Institut National Polytechnique de Toulouse (INP-ENSEEIHT)
Institut de Recherche en Informatique de Toulouse (IRIT)

Habilitation à Diriger les Recherches

Sécurité des réseaux et infrastructures critiques

Soutenance le 03 *décembre* 2009

Anas ABOU EL KALAM

Président du jury :	Danielle Boulanger, professeur Université de Lyon, Jean Moulin
Rapporteurs :	Fréderic Cuppens, professeur TELECOM Bretagne
	Maryline Laurent, professeur TELECOM SudParis
	Fatma Mili, professeur "Oakland University", Etats-Unis
Examinateur :	Yves Deswarte, directeur de recherche CNRS LAAS
	Christian Fraboul, professeur INPT
	Mohamed Mosbah, professeur Institut Polytechnique de Bordeaux
	Paolo Verìssimo, professeur "Universidade de Lisboa", Portugal

Table des Matières

Introduction générale

Le présent rapport présente une partie de mes travaux de recherche durant les cinq dernières années. Il se focalise sur mes recherches effectuées dans le cadre du projet européen CRUTIAL (*CRitical Utility InfrastructurAL resilience)* et du projet Airbus ADCN (*Advanced Data Communication Network*). C'est dans cette optique qu'il s'intéresse à la sécurité des réseaux et infrastructures critiques.

Ce sujet est d'actualité étant donné que l'informatisation (et avec elle les problèmes de sécurité) s'impose dans la quasi-totalité des réseaux et infrastructures critiques actuels et émergents : réseaux avioniques embarqués, réseaux satellites, réseaux de transport et de distribution d'électricité, etc. L'enjeu économique et social est désormais important ! Il devient donc de plus en plus nécessaire de faire face aux intrusions et d'avoir une confiance justifiée dans les traitements et la distribution des données et services. Ceci ne peut être garanti que si on fait appel à des méthodes et techniques pour la construction de réseaux et systèmes sûrs, mais aussi pour leur validation.

L'un des moyens pour avoir confiance dans les systèmes et réseaux est d'empêcher − *par construction* − l'occurrence ou l'introduction d'intrusions. La conception des politiques de sécurité en est un exemple où on spécifie les objectifs de sécurité à atteindre ainsi que les règles qui régissent la manière dont l'information sensible et les autres ressources sont gérées, protégées et distribuées. En général, les politiques de sécurité sont exprimées par des règles de permissions, interdictions et obligations. Ces politiques doivent bien évidemment être implémentées par des mécanismes appropriés : chiffrement, règles des pare-feux, listes de contrôles d'accès, certificats, etc.

Néanmoins, la définition d'une politique de sécurité ne garantit pas à elle seule le fonctionnement correct du système. Elle peut en effet être mal conçue ou violée intentionnellement (c'est-à-dire par malveillance) ou accidentellement (par exemple, une erreur de conception). Il est donc nécessaire de s'assurer du respect de cette politique (principalement par des méthodes et modèles formels) et d'en détecter les violations (principalement, par des méthodes de détection d'intrusion).

Dans ce contexte, mes travaux antérieurs et actuels sont centrés autour de moyens pour la prévention des intrusions (à travers la spécification de politiques de sécurité), mais aussi pour l'élimination des intrusions (à travers la vérification des modèles de sécurité et le test des outils de sécurité). Les thématiques que nous développons dans ce rapport sont les suivantes :

Modélisation de politiques de sécurité pour les infrastructures critiques : dans ma thèse et dans le projet MP6, j'ai contribué au développement du modèle OrBAC [Abou El Kalam *et al.*, 2003]. Celui-ci offre plusieurs avantages faisant de lui un modèle de contrôle d'accès en forte extension. Néanmoins OrBAC reste un modèle centralisé qui s'applique difficilement aux infrastructures et réseaux critiques qui nous intéressent (par exemple, le réseau européen de transport et de distribution d'électricité). En effet, par nature, ces systèmes et réseaux sont hétérogènes, distribués et impliquent plusieurs partenaires, parfois en concurrence. Pour couvrir ce type de besoins de sécurité, nous avons proposé le cadriciel (en anglais, *framework*[1]) *PolyOrBAC* ; celui-ci repose à la fois sur OrBAC, les mécanismes de

[1] Le terme cadriciel (ou *framework*) est utilisé ici pour désigner à la fois le modèle de sécurité et les mécanismes permettant son implémentation.

services Web et les contrats électroniques. Par ailleurs, en plus des permissions, obligations et interdictions, les politiques de sécurité actuelles incluent des règles de recommandations. Aussi, avons nous développé des travaux sur la modélisation du concept de recommandation.

Analyse de politiques de sécurité : la modélisation de la politique de sécurité ne peut être intéressante que si on est capable de raisonner logiquement sur cette politique et de s'assurer de sa cohérence. Ainsi, nous avons préconisé l'utilisation de la programmation logique par contraintes, d'abord pour spécifier les règles de fonctionnement du système ainsi que les règles de la politique de sécurité, mais aussi pour interroger la politique de sécurité et détecter et résoudre les conflits éventuels.

Mise en œuvre de politiques de sécurité et de qualité de service (QoS) : lors de notre étude des réseaux critiques, notamment les réseaux avioniques embarqués, nous avons identifié un fort besoin de sécurité et de qualité de service. En effet, pour ce type d'applications, il serait inacceptable de privilégier la sécurité au détriment de la QoS ou de proposer des mécanismes de sécurité qui ne respectent pas les contraintes temporelles. Pour répondre à ce besoin, nous avons proposé une amélioration du protocole IPSec qui assure la sécurité tout en permettant la mise en œuvre des mécanismes de QoS. Notre nouveau protocole, baptisé Q-ESP (*QoS-friendly Encapsulating Security Payload Protocol*) a été spécifié, implémenté et évalué analytiquement et expérimentalement.

Évaluation des outils de sécurité : si les modèles formels de sécurité servent essentiellement à vérifier la cohérence et la complétude de la politique de sécurité, il devient parfois nécessaire de s'assurer - lors du fonctionnement du système - (en mode opérationnel) que les outils et mécanismes de sécurité se comportent comme prévu, en particulier en détectant et/ou empêchant toute violation des politiques de sécurité. Nous nous sommes ainsi intéressés au test de la robustesse, de l'efficacité et des performances des outils de sécurité tels que les pare-feux ou les systèmes de détection d'intrusions (IDS, pour *Intrusion Detection Systems*). Dans ce sens, nous avons développé une méthode systématique pour l'évaluation, les modèles ainsi que les tests nécessaires. Un outil (*plugin* pour le *framework metasploit*) pour la planification et l'exécution des tests a été développé pour cela.

Développant ces thématiques, le reste du présent rapport sera structuré comme suit :

Le premier chapitre dérive progressivement vers un modèle de contrôle d'accès pour l'expression de politiques de sécurité des systèmes collaboratifs en général et des infrastructures critiques en particulier. Partant d'OrBAC dont nous mettons en avant les atouts et les limites, nous présenterons le *framework* PolyOrBAC ainsi que nos derniers travaux sur la modélisation des recommandations.

Les modèles de contrôle d'accès servent non seulement à spécifier les politiques de sécurité, mais aussi à pouvoir raisonner mathématiquement sur de telles politiques. Ainsi, le deuxième chapitre vient présenter nos travaux sur la détection de conflits dans les politiques de sécurité.

Le troisième chapitre passe du stade de la modélisation à celui de la réalisation. En effet, nous y présentons notre nouveau protocole de contrôle d'accès qui étend IPSec pour mettre en œuvre à la fois des politiques de sécurité et de QoS, deux besoins essentiels dans les réseaux critiques, notamment les réseaux à contraintes temporelles tels que les réseaux avioniques.

Notons toutefois que le déploiement de politiques de sécurité (notamment à travers des modules VPN ou règles de pare-feux) ne garantit pas forcément une implémentation saine de cette politique. Le quatrième chapitre s'intéresse ainsi aux tests et à l'évaluation des outils de sécurité. Nous nous concentrons en particulier sur des architectures de sécurité incluant des IDS.

Enfin, nous terminons ce rapport par nos conclusions et perspectives de recherche.

Chapitre 1 : Expression de politiques de sécurité pour les infrastructures critiques

I. Introduction

Afin d'éliminer les vulnérabilités et de contrer les attaques, on fait généralement appel à des services, mécanismes, outils et procédures que l'on nomme communément solutions ou mesures de sécurité. Les politiques de sécurité en sont un composant incontournable si on souhaite suivre une démarche rigoureuse et garantir un niveau élevé de protection.

Plusieurs définitions des politiques de sécurité peuvent être trouvées dans la littérature. Par exemple, les critères communs les définissent comme étant « *l'ensemble des règles, procédures et directives de sécurité imposées dans l'environnement opérationnel (actuel et/ou futur) par une organisation (hypothétique ou réelle)* [Common Criteria 2006]. De manière concrète, la conception d'une politique de sécurité consiste (1) à identifier les objectifs de sécurité que le système doit satisfaire (par exemple, *les soumissions à des appels d'offres ne doivent pas être divulguées à des organismes concurrents*), et (2) à élaborer les règles régissant l'état de protection du système, par exemple comment les informations sensibles seront protégées, comment les droits peuvent être propagés (e.g., *le propriétaire d'une information peut accorder un droit d'accès pour cette information à n'importe quel utilisateur*).

Une politique de sécurité se développe selon trois axes : physique, administratif et logique. Le premier précise l'environnement physique du système à protéger (e.g., les mesures prises vis-à-vis du vol et des catastrophes). Le deuxième décrit les procédures organisationnelles (répartition des tâches, formation des utilisateurs). Le troisième a trait aux contrôles d'accès logiques (qui a accès à quoi, quand, dans quelles conditions ?) Et s'intéresse ainsi aux fonctions d'identification, d'authentification et d'autorisation mises en œuvre par le système informatique. Dans ce rapport, nous nous intéressons particulièrement aux politiques d'autorisation, dites aussi politiques de contrôle d'accès.

En général, la politique de contrôle d'accès est associée à un modèle de contrôle d'accès. Celui-ci peut être vu comme un formalisme permettant de représenter cette politique de façon claire et non-ambiguë. Il aide à l'abstraire (afin de réduire sa complexité) et à faciliter sa compréhension, comme il peut servir à vérifier que cette politique est complète et cohérente, et que la mise en œuvre par le système de protection est conforme aux propriétés attendues du système.

Il est donc évident que pour proposer des modèles et politiques de contrôle d'accès adaptés à un certain système, il faut tout d'abord avoir une connaissance précise de ses besoins de sécurité.

Le but de ce chapitre est de développer des politiques et modèles de sécurité pour les systèmes interopérables, hétérogènes, avec des accès intra et inter-organisationnels, contraintes qu'on trouve notamment dans les infrastructures critiques.

La section suivante décrit les besoins de sécurité des infrastructures critiques (IC). Ensuite, nous présentons le modèle OrBAC sur lequel nous nous sommes basés pour mener cette étude. Enfin, nous présentons PolyOrBAC, notre extension d'OrBAC qui couvre la richesse des systèmes qui nous intéressent.

Nous avons fait le choix de commencer notre étude des politiques de sécurité par l'analyse des besoins des IC pour plusieurs raisons :

- les IC sont au cœur du projet européen CRUTIAL auquel nous avons participé [Abou El Kalam *et al.*, 2007b, Verìssimo *et al.* 2008] ;
- les IC possèdent des exigences fonctionnelles variées ainsi que des contraintes à la fois fortes et représentatives de sécurité ;
- nos relations avec des professionnels du domaine qui nous ont aidé à valider nos scénarios et nos résultats.

Toutefois, le fruit de ce travail, PolyOrBAC, reste assez générique et peut s'appliquer à tout type de systèmes qui ont plus ou moins les mêmes contraintes que les IC.

II. Besoins de sécurité des infrastructures critiques

Une **I**nfrastructure **C**ritique (**IC**) est constituée d'un ensemble d'organisations, installations, équipements, biens logiques et physiques, qui a une importance vitale ou critique pour le fonctionnement d'une économie en particulier ou de la société humaine en général, et dont la défaillance, l'arrêt de fonctionnement ou la dégradation peuvent avoir un impact potentiellement dramatique sur le bien-être économique et social d'une nation [Moteff & Parfomak 2006, Moteff *et al.*, 2003]. Globalement, trois types d'infrastructures critiques peuvent être identifiés : infrastructures d'approvisionnement qui regroupent entre autres les fournitures d'eau, de gaz et de carburant, infrastructures tertiaires qui regroupent entre autres les services bancaires, sanitaires, sécuritaires (e.g., police, armée), et enfin les infrastructures de transport aérien, terrestre et maritime.

Le fonctionnement d'une IC dépend bien évidement du fonctionnement du système d'information et de communication qui lui est lié, celui-ci est nommé "**I**nfrastructure d'**I**nformation **C**ritique (IIC)". Dans ce rapport, nous nous concentrerons en particulier sur la sécurité des IIC dédiées à la grille d'énergie électrique.

Alors que ces infrastructures étaient fermées et n'impliquaient que peu d'intervenants, des facteurs tels que la mondialisation, le besoin de partage de données et services, de communication, et de mutualisation des coûts, ont mené ces IIC à intégrer des technologies plus ou moins vulnérables (par exemple, connexion à certains réseaux à base d'IP, utilisation de systèmes d'exploitation grand public, intégration de logiciels sur étagères, etc.). Ces vulnérabilités sont naturellement confrontées à des menaces spécifiques pouvant mener à des pannes importantes, par exemple un dysfonctionnement du système suite à une mauvaise manipulation accidentelle, ou plus grave encore, une malveillance (attaque logique par un groupe de terroristes).

Afin de pallier ces risques et garantir un niveau élevé de protection du réseau et du système d'information, la politique de sécurité doit tenir compte d'un certain nombre de caractéristiques

conceptuelles et fonctionnelles des IIC, notamment [Moteff & Parfomak 2006, [Abou El Kalam & Deswarte 2009a] :

- **Complexité :** afin d'assurer des besoins opérationnels, une IIC contient des systèmes logiques et physiques complexes et spécifiques à plusieurs domaines tels que les systèmes de production, transport et distribution de l'énergie électrique dans le réseau d'énergie électrique.

- **Extensibilité en taille et structure :** les changements continuels de structure et l'aspect dynamique de l'IIC imposent des propriétés de souplesse, flexibilité et extensibilité en taille, structure, nombre d'utilisateurs et de composants.

- **Caractère multi-organisationnel :** une IIC interconnecte plusieurs organisations participant à une même IC ou appartenant à différentes IC.

- **Les accès distants :** une IIC peut s'étendre sur plusieurs zones géographiques, potentiellement éloignées (par exemple, le réseau électrique européen qui relie entre-autres, la France, la Belgique, l'Italie et l'Espagne) et dont certains services et données peuvent être accessibles par des utilisateurs externes, voir même appartenant à d'autres pays.

- **Interdépendance :** l'inter-connectivité peut induire une interdépendance entre les différents composants. Le fonctionnement d'une organisation a potentiellement des conséquences sur les autres organisations qui lui sont reliées, ce qui peut soulever des problèmes de sécurité qui doivent être résolus à la fois localement par chaque organisation, et au niveau de l'IIC par une gestion conjointe de la sécurité globale.

- **Interopérabilité:** il est souhaitable que les organisations, interconnectées et interdépendantes utilisent des standards communs notamment pour la communication et le partage de données.

- **Concurrence et suspicion:** malgré l'intérêt commun à coopérer, les différentes organisations peuvent être en concurrence, et donc se méfier les unes des autres en raison des intérêts financiers en jeu, les mêmes qui les ont poussées à chercher des moyens pour coopérer. C'est le cas en particulier en Europe, où des entreprises régionales, nationales ou multinationales sont en concurrence mais doivent coopérer pour produire, transporter et distribuer l'énergie électrique.

- **Autonomie :** pour réduire l'interdépendance avec des organisations dont on se méfie, il est important que chaque organisation soit autant que possible autonome dans la gestion de ses ressources et de ses personnels. Du point de vue de la sécurité, cela signifie que chaque organisation peut choisir en toute indépendance ses moyens d'authentification ainsi que ses mécanismes de contrôle d'accès. Les interactions avec d'autres organisations ne doivent rien révéler de la structure interne de l'organisation (ressources, utilisateurs, etc.).

La question qui se pose maintenant est comment peut-on établir des politiques de sécurité qui couvrent la richesse des besoins cités précédemment ? Notre analyse nous a conduit à élaborer un nouveau cadriciel qui soit le plus évolutif, expressif, flexible, structuré et contextuel ; nous l'avons baptisé Poly-OrBAC (pour *Poly-Organization based Access Control*) [Abou El Kalam *et al.*, 2009, [Abou El Kalam & Deswarte, 2009b]. Celui-ci est basé sur OrBAC (pour *Organization based Access Control*) au niveau du modèle de contrôle d'accès [Abou El Kalam *et al.*, 2003]. OrBAC puis PolyOrBAC sont respectivement décrits dans les deux sections suivantes.

III. Le modèle OrBAC

A. *Concepts clés du modèle*

Les toutes premières représentations des politiques de sécurité utilisaient les matrices de contrôle d'accès [Lampson 1971, Harrison *et al.* 1976]. Chaque cellule *M*(*s,o*) de cette matrice contient les droits d'accès (e.g., lire, écrire) que le sujet *s* possède sur l'objet *o*. La matrice des droits d'accès n'est pas figée. En effet, si de nouveaux objets, sujets ou actions sont ajoutés dans le système, il devient alors nécessaire d'enregistrer toutes les permissions accordées pour ces nouvelles entités. Par conséquent, la mise à jour d'une politique de sécurité exprimée par ce modèle est quelque peu fastidieuse.

A l'inverse, les politiques basées sur les rôles (RBAC, pour *Role-Based Access Control* [Sandhu *et al.* 2000 ; Solms & Merwe 1994], pour ne citer que quelques-uns) visent à faciliter l'administration de la sécurité en introduisant la notion de *rôle*. Un rôle représente de façon abstraite une fonction identifiée dans l'organisation (par exemple, chef de service, ingénieur). Dans RBAC, les permissions ne sont plus associées d'une façon directe aux sujets, mais à travers des rôles. La politique de sécurité est donc décrite à travers deux relations : "*Détient*(*Rôle*, *Permission*)" et "*Joue*(*Sujet*, *Rôle*)". De cette manière, les sujets acquièrent les permissions à travers les rôles qu'ils jouent à un moment donné.

Le modèle OrBAC pousse ce raisonnement en séparant complètement l'expression de la politique de sécurité de son implémentation, et en structurant, au sein des organisations, non seulement les sujets (à travers les rôles), mais aussi les objets (à travers les vues) et les actions (à travers les activités). En effet, de même qu'un rôle (e.g., médecin) est une représentation abstraite d'un groupe d'utilisateurs exerçant une fonction dans une organisation (e.g., Alice et Bob qui jouent le rôle médecin), une activité (e.g., lecture) représente de manière abstraite une ou plusieurs actions (e.g., les fonctions implémentant l'activité de lecture comme *Openf*(), *Select*, *OpenXmlFile*()) et une vue (e.g., dossier médical) correspond à un ensemble d'objets (e.g., les fichiers *F_DM.xml* et la table *T_DM* de la base de donnée des dossiers médicaux).

Afin d'associer les entités abstraites (rôles, vues et activités) aux entités concrètes (sujet, objet, action) d'une organisation donnée, OrBAC introduit les relations *Habilite*(), *Utilise*() et *Considère*(). Par exemple :

- *Habilite*(*Purpan*, *Bob*, *cardiologue*) signifie que l'hôpital *Purpan* habilite *Bob* dans le rôle *cardiologue*.
- Utilise(*Purpan*, *F_DM.odt*, *dossier_médical*) signifie que *Purpan* utilise le fichier *F_DM.odt* comme un dossier médical, et *Utilise*(*Rangueil*, *Table*1*_DM*, *dossier_médical*) signifie que l'hôpital *Rangueil* utilise la table relationnelle *Table*1*_DM* comme un dossier médical.
- *Considère*(*Purpan*, *Openf*(), *consultation*) signifie que *Purpan* considère la fonction "*Openf*()" comme une action de l'activité consultation, et *Considère*(*Rangueil*, *select*, *consultation*) signifie que *Rangueil* considère "*select*" comme une action de l'activité consultation.

OrBAC définit également une notion de contexte comme une situation spécifique qui conditionne la validité d'une règle. Autrement dit, une règle n'est pas tout le temps applicable, mais son activation dépend de la satisfaction, en temps-réel, de certaines conditions. De ce fait, OrBAC est un modèle de contrôle d'accès dynamique qui gère notamment les autorisations contextuelles. Celles-ci peuvent dépendre de paramètres tels que le temps (interdire ou permettre une certaine action selon les heures de

travail), la localisation (permission ou interdiction selon l'adresse de provenance de la requête d'accès), ou l'historique des actions passées (permission si d'autres actions ont été préalablement réalisées).

Afin de modéliser le contexte, OrBAC introduit la relation *Définit()* qui relie un sujet, un objet et une action au sein d'une organisation donnée. Par exemple, *Définit*(*Purpan*, *Bob*, *openf*(), *F*31.*odt*, *urgence*) et *Définit*(*Rangueil*, *Marie*, *Select*, *Table*1, *médecin_traitant*) [Cuppens & Cuppens-Boulahia 2008].

OrBAC distingue ainsi deux niveaux : (1) un niveau organisationnel où la politique OrBAC est exprimée par des entités abstraites (rôles, vues et activités) sans s'inquiéter de la façon dont l'organisation implémente ces entités, et (2) un niveau concret qui correspond aux règles dérivées (dans un contexte donné, pour un sujet, objet et action donnés) de la politique organisationnelle. Dans ce niveau, il s'agit d'entités actives (typiquement, des processus informatiques) qui exécutent des actions sur des objets, sous le contrôle de mécanismes de protection qui mettent en œuvre les règles définies dans la politique.

OrBAC propose aussi une représentation formelle liant tous ces concepts et permettant de dériver les décisions d'accès (règles au niveau concret) à partir des règles du niveau organisationnel et du contexte courant comme indiqué dans le Tableau 1 :

Tableau 1 : Définition d'une permission avec le formalisme OrBAC.

$\forall org \in Organisations, \forall s \in Sujets, \forall \alpha \in Actions, \forall o \in Objets, \forall r \in Rôles, \forall a \in Activités, \forall v \in vues, \forall c \in Contextes$

$$Permission(org, r, v, a, c) \wedge Habilite(org, s, r) \wedge Considère(org, \alpha, a) \wedge Utilise(org, o, v) \wedge Définit(org, s, \alpha, o, c) \rightarrow Est\_permis(s, \alpha, o)$$

Cette formule s'interprète de la façon suivante : si dans l'organisation *org*, le rôle *r* est autorisé à effectuer l'activité *a* sur la vue *v* quand le contexte *c* est vrai, et si dans *org*, le rôle *r* est assigné au sujet *s*, l'action *α* fait partie de l'activité *a*, l'objet *o* fait partie de la vue *v*, le contexte *c* est vrai pour le quadruplet (*org*, *s*, α, *o*), alors le sujet *s* a le droit d'exécuter l'action α sur l'objet *o*.

Par ailleurs, de la même manière qu'OrBAC définit des règles de permissions, il définit des règles d'interdictions, d'obligations et de recommandations. Par exemple, $Interdiction(org, r, v, a, c) \wedge Habilite(org, s, r) \wedge Considère(org, \alpha, a) \wedge Utilise(org, o, v) \wedge Définit(org, s, \alpha, o, c) \rightarrow Est\_Interdit(s, \alpha, o)$.

B. *OrBAC : atouts et étude de complexité*

Les idées introduites dans OrBAC (e.g., séparation complète de la politique de sécurité de son implémentation) apportent plusieurs avantages tant sur le plan conceptuel que sur le plan pratique. En particulier :

- Aider l'administrateur à définir la politique de sécurité (au niveau abstrait) sans se soucier ni des

implémentations courantes ni de leurs évolutions possibles. En effet, les fréquents changements dans les implémentations et plates-formes, par exemple l'ajout d'un nouvel objet, ne devraient pas induire un changement dans la politique de sécurité au niveau organisationnel.

- Une même règle de sécurité au niveau abstrait peut être instanciée différemment (selon l'organisation concernée, l'implémentation courante, etc.) et donner ainsi naissance à plusieurs règles dérivées. Ceci réduit la complexité de la politique de sécurité et la rend plus reproductible, évolutive et dynamique. De plus, les règles OrBAC sont applicables aussi bien dans un contexte système que réseau [Cuppens *et al.* 2004].

- Réduire les erreurs ; en effet un changement de la politique globale ne nécessite aucun réajustement au niveau concret qui pourrait introduire des incohérences, souvent difficiles à repérer et à corriger.

Dans le reste de cette section, nous démontrons que malgré sa richesse, OrBAC n'introduit pas une complexité supplémentaire ; au contraire, il réduit les risques d'erreurs et améliore le coût de gestion de la politique de sécurité. Pour démontrer cela, nous avons comparé OrBAC avec les modèles RBAC et Lampson selon trois points essentiels [Abou El Kalam 2008b] :

- la complexité des opérations de prise de décision de contrôle d'accès (décider si une action est autorisée ou pas),

- le coût des mises à jour de la politique de sécurité, par exemple pour changer les droits d'un utilisateur,

- les risques d'erreurs lors de l'administration de la politique de sécurité.

Durant notre étude, nous adoptons la notation suivante :

- *Organisations*, *Sujets*, *Actions*, *Objets*, *Rôles*, *Activités* désignent les ensembles des entités organisations, sujets, actions, objets, rôles et activités.

- |.| désigne le cardinal (nombre d'éléments) ; par exemple, |*Rôles*| est le nombre possible de rôles.

- $\boldsymbol{n} = Max$ (|*Sujets*|, |Objets|).

Nos hypothèses de départ sont les suivantes :

- |*Sujets*| et |*Objets*| sont potentiellement plus élevées que |*Organisations*|, |*Rôles*|,|*Activités*| et |*Vues*|, ce qui est le cas de la plupart des applications réelles.

- Les opérations pouvant avoir un impact sur tout ou une partie de la politique de sécurité (impliquant plusieurs utilisateurs et/ou objets) sont qualifiées de sensibles et donc effectuées par l'administrateur ; à l'inverse les opérations ayant un impact relativement limité sont qualifiées de moins sensibles et peuvent être déléguées à une personne qui n'a pas nécessairement une vue conceptuelle globale sur la politique de sécurité, par exemple un opérateur ou une secrétaire.

Notons que nos études de complexité sont complémentaires à celles présentés dans [Cuppens-Boulahia 2007]. Celles-ci utilisent plutôt l'algèbre des contextes ainsi qu'une stratégie hybride (ascendante et descendante) pour garantir la décidabilité de l'évaluation des requêtes OrBAC et sa terminaison en temps polynomial [Toman 1997].

a. Coûts des décisions de contrôle d'accès

Intuitivement, l'étude des coûts des décisions de contrôle d'accès prend en compte deux paramètres importants : le nombre de décisions et le coût de chaque décision. En effet, un grand nombre d'opérations complexes implique naturellement plus de ressources, plus de temps de traitement et éventuellement une probabilité d'erreurs plus élevée. Comme précisé dans nos hypothèses, deux types d'opérations de gestion peuvent être distinguées dans notre contexte :

- opérations sensibles impliquant la vision globale de la politique de sécurité et qui doivent donc être faites par l'administrateur sécurité. Dans OrBAC, il s'agit des règles du type *Permission* (*org*, *r*, *v*, *a*, *c*). Notons ***D*** le coût de telles opérations.

- opérations moins sensibles qui peuvent être gérées par la secrétaire ou l'opérateur ; notons ***d*** le coût de telles opérations. Dans OrBAC, il s'agit des opérations d'affectation des sujets aux rôles, objets aux vues et actions aux activités.

Dans le modèle de Lampson, le nombre total des opérations de décision de contrôle d'accès est : |Sujets|.|Objets|.|Actions| = |Sujets|.|Objets|.|*constante*|. = $O(n).O(n).O(1) = O(n^2)$.

Par ailleurs, comme toutes ces opérations nécessitent l'intervention de l'administrateur, la complexité est paramétrée par ***D***. Le coût total est donc $\boldsymbol{D}.O(n^2)$.

Dans RBAC, la relation d'association des rôles aux sujets est moins sensible que celle associant les permissions aux rôles. Le coût est donc :

$d.|Joue(Sujet, Rôle)| + D.|Détient(Rôle, Permission)| = d.|Sujets|.|Rôles| + D.|Rôles|.|Actions|.|Objets|$

$= d.|Sujets|.|constante| + D.|constante|.|constante|.|Objets|$

$= d.O(n) + D.O(n)$

Dans le cas d'OrBAC, étant donné que la décision d'accès dépend des règles *Permission* (*org*, *r*, *v*, *a*, *c*) $\wedge$ *Habilite* (*org*, *s*, *r*) $\wedge$ *Considère* (*org*, α, *a*) $\wedge$ *Utilise* (*org*, *o*, *v*) $\wedge$ *Définit* (*org*, *s*, α, *o*, *c*) $\rightarrow$ *Est_Permis*(*s*, α, *o*), le coût est :

$\boldsymbol{D}.|Règle\_Abstraite| + \boldsymbol{d}.[\ |Habilite()| + |Considère()| + |Utilise()|\ ]$, où

$|Règle\_Abstraite| = |Modalité\_d\_accès| + |Organisations| + |Rôles| + |Activités| + |Vues|$

$= |constant|+|constant|+|constant|+|constant|+|\ constant|$

$\approx O(1)$

$|Utilise()| = |Organisations| + |Vues| + |\text{Objets}|$

$\approx |constant| + |\ constant\ | + O(n)$

$\approx O(n)$

Suivant le même raisonnement, $|Habilite()| \approx O(n)$ et $|Considère()| \approx O(1)$

Ainsi,

$Coût \approx \boldsymbol{D}.\ O(1) + \boldsymbol{d}\ .\ [O(n) + O(1) + O(n)\ ]$

$\approx \boldsymbol{D}.\ O(1) + \boldsymbol{d}.O(2n)$

Le tableau 2 récapitule le coût des décisions d'accès pour chacun des trois modèles.

Tableau 2. coût des décisions d'accès dans Lampson, RBAC et OrBAC.

Lampson	RBAC	OrBAC
$D.O(n^2)$	$D.O(n) + d.O(n)$	$d.O(2n) + D.\ O(1)$

Dans ce tableau, on peut voir que la complexité pour Lampson est une fonction quadratique avec un facteur élevé (D), tandis que dans RBAC, il s'agit d'une fonction linéaire avec deux facteurs, l'un majeur (D) et l'autre mineur (d). En effet, en utilisant les rôles (au lieu des sujets), RBAC réduit le coût des prises de décision d'accès. OrBAC va plus loin en structurant les objets et les actions ; sa complexité est une fonction linéaire avec un facteur mineur seulement.

b. Mise à jour de la politique de sécurité

Dans le modèle de Lampson, la gestion des permissions (ajout, mise à jour, suppression) d'un sujet est de l'ordre de $O(n)$. De plus, toutes ces opérations nécessitent une décision majeure. Le coût total est donc $D.O(n)$, étant donné qu'il s'agit d'opérations sensibles qui nécessitent de revoir / parcourir toutes les permissions des sujets.

A l'inverse, dans RBAC et OrBAC, la gestion des permissions d'un sujet correspond en fait au changement de ses rôles. Dans les deux cas, ce type de gestion est de $O(1)$ opérations seulement.

Si on s'intéresse maintenant au changement des permissions associées à un certain objet, le coût est $D.O(n)$ (comme pour les sujets) pour Lampson. Dans RBAC, le coût est de l'ordre de $D.O(1)$, étant donné qu'on parcourt les instances de la relation *Détient(Rôle, Permission)*. Enfin, dans OrBAC, le coût est de l'ordre de $d.O(1)$ car la seule relation considérée est *Considère*(). Tableau 3 récapitule ces résultats.

Tableau 3. Complexité des mises à jour des règles de contrôle d'accès.

Lampson	RBAC	OrBAC
$D.O(n)$	$D.O(1)$	$d.O(1)$

Ainsi, alors que Lampson a une complexité linéaire, et que RBAC a une complexité constante avec un facteur majeur, OrBAC n'a qu'une complexité constante avec un facteur mineur.

c. Risques d'erreurs d'administration de la politique

Pour évaluer ce type de risques, nous utilisons deux indicateurs :

- la sévérité de la menace, estimée en terme de son impact, et
- la potentialité de la menace, estimée en terme de sa probabilité (si la menace est accidentelle) ou de sa fréquence (si elle est volontaire).

Comme expliqué précédemment, les décisions de l'administrateur sécurité sont potentiellement plus sensibles que celles de l'opérateur ou de la secrétaire. Dans le premier cas, nous notons la sévérité $\boldsymbol{S}$, tandis que dans le deuxième, nous la notons $\boldsymbol{s}$.

Par ailleurs, la potentialité (e.g., probabilité) qu'un administrateur fasse une erreur reste inférieure à celle de l'opérateur. Notons $\boldsymbol{P}$ la potentialité d'une erreur de l'administrateur, et $\boldsymbol{p}$ la potentialité d'une erreur de l'opérateur.

Par conséquent, le facteur de risque (d'erreurs) des opérations de l'opérateur est équivalant à ***s.P***, tandis que pour l'administrateur est de ***S.p***.

Ainsi, pour calculer le risque d'erreurs pour les trois modèles comparés, il suffit de remplacer ***D*** (de Tableau 2) par ***S.p*** et ***d*** par ***s.P***. Par exemple, pour le modèle de Lampson, comme les décisions de gestion du contrôle d'accès sont de l'ordre de $O(n^2)$ (voir Tableau 2), et comme toutes ces opérations sont supposées être faites par l'administrateur, le risque d'erreurs est de l'ordre de $S.p.O(n^2)$. Tableau 4 résume nos résultats de comparaison.

Tableau 4. Les risques d'erreurs d'administration dans Lampson, RBAC et OrBAC.

Lampson	RBAC	E-RBAC
$S.p.O(n^2)$	$S.p.O(n) + s.P.O(n)$	$s.P.O(2n) + S.p.O(1)$

On peut donc conclure que non seulement OrBAC gagne en clarté, évolutivité, dynamicité et expressivité, mais aussi réduit les coûts de gestion de la politique de sécurité ainsi que les risques de son administration. OrBAC constitue donc un bon candidat pour la suite de nos travaux dans le domaine des politiques et modèles de sécurité.

IV. Le cadriciel PolyOrBAC

A. *Motivation*

Dans la Section II, nous avons fixé le contexte des applications qui nous intéressent, en détaillant les besoins et les exigences de sécurité qu'on trouve notamment dans les infrastructures critiques : complexité, extensibilité, accès dynamiques inter- et intra-organisationnel, interdépendance, interopérabilité, autonomie, concurrence et suspicion. Ensuite, dans la Section III, nous avons présenté le modèle OrBAC ainsi que les atouts qui font de lui un modèle intéressant : maîtrise de la complexité, dynamicité, accès contextuels, évolutivité et extensibilité.

Dans cette section, nous allons présenter notre extension d'OrBAC qui vise à couvrir la richesse des systèmes distribués avec un fort besoin d'interactions entre des services, composants et utilisateurs appartenant à des organisations distantes parfois en concurrence, comme c'est le cas notamment des IC.

OrBAC nous semble utile et puissant pour exprimer les politiques locales de contrôle d'accès de chacune des organisations des IIC. Sa capacité d'abstraction permet de spécifier dans un format unique des règles de sécurité de plusieurs systèmes hétérogènes, tout en offrant un fonctionnement adaptable et flexible pour chaque organisation. Ceci facilite ainsi la spécification de politiques de sécurité d'organisations plus ou moins complexes, sans entrer dans le détail des utilisateurs et des ressources qu'elles doivent contrôler. De plus, la définition récursive du concept d'organisation permet de maîtriser la complexité en généralisant la notion d'héritage des organisations, rôles, vues et activités. Nos travaux présentés dans [Abou El Kalam & Ouabiba 2005, Abou El Kalam & Deswarte 2006] donnent plus de détails sur l'héritage dans OrBAC. La séparation entre la spécification et l'implémentation de la politique de sécurité OrBAC rend cette dernière plus facilement extensible. Enfin, la prise en compte du contexte dans OrBAC lui permet d'exprimer des politiques de sécurité plus riches et plus dynamiques.

Néanmoins, OrBAC reste un modèle centralisé qui gère mal la collaboration entre des organisations non-hiérarchiques. Il ne spécifie pas comment exprimer les accès externes qui impliquent plusieurs organisations autonomes ; autrement dit, il ne précise pas comment associer des permissions à des utilisateurs appartenant à des organisations distantes autonomes et indépendantes. Bien évidement, avec OrBAC, on pourrait imaginer un schéma hiérarchique où on dispose d'une organisation mère dans laquelle on définit une politique globale qui s'imposerait à tous, puis de raffiner la spécification en décrivant des politiques locales au niveau de chacune des organisations. Toutefois, outre le fait que définir une telle politique globale serait un exercice difficile (dès que le système est d'une complexité significative), ceci serait nécessairement en contradiction avec les besoins d'autonomie des organisations, d'évolutivité et d'extensibilité des infrastructures. De plus, la vérification de la cohérence des politiques de sécurité, mais surtout entre ces politiques et (au moins) celles des super- et sous-organisations, serait certainement une tâche fastidieuse. Ceci est d'autant plus complexe qu'il faut revérifier la cohérence à chaque fois qu'il y a une modification dans une des politiques.

Pour couvrir tous les besoins des systèmes qui nous intéressent, nous avons proposé le cadriciel PolyOrBAC [Abou El Kalam *et al.*, 2007a], [Abou El Kalam & Deswarte, 2009], [Abou El Kalam *et al.*, 2009]. De manière globale, PolyOrBAC permet de spécifier, déployer et d'effectuer certaines vérifications sur une politique de sécurité gérant des accès à la fois inter- et intra-organisationnels. Il permet de répondre à des questions du type : comment spécifier les accès inter-organisationnels ? Comment spécifier et déployer les contrats électroniques établis entre les organisations qui collaborent ? Comment peut-on appliquer la politique de sécurité, vérifier et détecter les abus et violations potentiels de cette politique ? Pour arriver à ses fins, PolyOrBAC se base sur trois composants principaux :

- le modèle de sécurité OrBAC, utilisé pour spécifier la politique de sécurité locale à chaque organisation, ce modèle étant étendu pour prendre en compte les mécanismes de coopération avec les autres organisations [Abou El Kalam *et al.* 2003] ;
- la technologie des services Web qui nous permet de fournir une plate-forme de collaboration et d'interopérabilité entre les organisations, avec des extensions pour permettre de spécifier et de mettre en œuvre la sécurité sur les interactions [Abou El Kalam *et al.*, 2007a, Abou El Kalam & Deswarte 2009b] ;
- les mécanismes de vérification par modèle dits de *model-checking* pour la vérification automatique du respect des contrats électroniques établis entre les organisations qui collaborent, ce qui peut être intéressant pour la vérification de certaines propriétés à assurer (e.g., sûreté, vivacité), mais aussi pour détecter les abus dans le cas d'accès externes entre des organisations en suspicion [Abou El Kalam & Deswarte, 2009a, Abou El Kalam *et al.*, 2008a].

Ces composants ainsi que la façon dont ils sont gérés au sein de PolyOrBAC seront présentés dans la section suivante.

B. *PolyOrBAC*

a. Gestion des interactions dans PolyOrBAC

Comme cité précédemment, les politiques locales de chaque organisation sont exprimées avec OrBAC. Par ailleurs, afin de gérer l'interopérabilité ainsi que la collaboration et les différentes

interactions entre les organisations, nous nous sommes inspirés de l'Architecture Orientée Services, dite SOA. L'idée qui nous a séduit dans SOA est sa capacité de construire des applications hétérogènes et réutilisables en combinant des services interopérables à couplage lâche (*loose coopling*). Avec SOA, les services implémentés dans différents langages et s'exécutant sur des plate-formes hétérogènes peuvent échanger des données à travers le réseau de manière similaire aux communications inter-processus, ce qui permet de gérer ainsi l'interopérabilité et l'hétérogénéité (en masquant, entre autres, les détails et opérations liés à aux communications distantes).

La technologie des services Web utilise la notion de service de SOA et l'implémente avec un certain nombre de protocoles et normes tels que : *XML* (*Extensible Mark-up Language*) [W3C, 2004], un métalangage qui définit des *formats* pour décrire les données et faciliter leur transmission, interprétation et partage ; *SOAP* **(***Simple Object Access Protocol***)** [W3C, 2003], un mécanisme de *transport* (de données) afin d'échanger des données entre des applications s'exécutant sur différentes plate formes ; *WSDL* (*Web Services Description Language*) [W3C, 2006], un langage basé sur XML et utilisé pour *décrire les interfaces* des services que les prestataires offrent aux clients (e.g., type de paramètres nécessaires au moment de l'appel du service, type des valeurs retournées par le service) ; et *UDDI* (*Universal Description, Discovery, and Integration*) [OASIS, 2005], un *annuaire* (registre de publication et de localisation des services) basé sur XML, qui permet à des organisations éloignées, de lister et faire connaître leurs services sur Internet.

L'ensemble de ces outils permet de séparer la description abstraite des fonctionnalités offertes par un service des détails concrets du fonctionnement du service, notamment son implémentation (côté organisation prestataire) ainsi que sa localisation et la façon de l'appeler (côté organisation cliente). Cette manière de faire semble intéressante dans notre contexte, étant donné que notre but est de permettre à une organisation (et à ses utilisateurs) d'accéder de manière sécurisée aux ressources appartenant à d'autres organisations de l'infrastructure, sans pour autant divulguer les détails d'implémentation et de structure.

Toutefois, si l'interopérabilité et l'interconnexion entre organisations peuvent être gérées par les mécanismes des Services Web (SW), les problèmes liés à leur interfaçage avec les politiques locales exprimées en OrBAC ainsi que la "sécurisation" des accès inter-organisationnels restent à résoudre. Autrement dit, il faut mettre en place des mécanismes de contrôle d'accès au-dessus (sur) les interactions entre organisations, donc au niveau des SW, en fonction d'une politique de contrôle d'accès négociée entre les organisations collaborantes. Il faut aussi que la politique locale à l'organisation cliente (utilisant le SW) prenne en compte les SW fournis par l'organisation prestataire (fournissant le SW). Pour cela, et à l'instar des notions de *proxies* de CORBA et de *stub*/*skeleton* de la technologie RMI (*Remote method invocation*) [CORBA 98, Sun 2006, Hou & Xia 2009]; nous introduisons deux notions essentielles pour "virtualiser" (ou simuler) localement les objets et utilisateurs distants : *images de services* Web et *utilisateurs virtuels*. L'image de service Web est utilisée (au niveau de l'organisation cliente) pour représenter localement le service distant à invoquer ; tandis que la notion d'utilisateur virtuel est utilisée (au niveau de l'organisation qui fournit le service) pour représenter une organisation cliente (Figure 1). Ceci nous permet donc de simuler les accès distants comme s'ils étaient locaux.

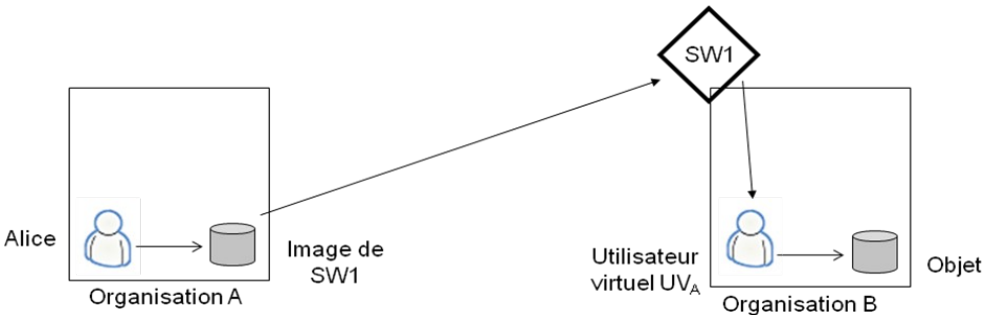


Figure 1 : Image de service Web et utilisateur virtuel.

Dans la pratique, nous distinguons deux grandes étapes dans PolyOrBAC : (1) Publication et négociation des règles de collaboration et spécification des règles de contrôle d'accès qui leurs sont liées, et (2) Accès en temps réel (à l'exécution) aux services distants.

Intéressons nous à la première étape et prenons un exemple simple où une organisation *B* décide d'ouvrir les accès à certaines de ses ressources vers l'extérieur. *B* crée donc le Service Web qui permet cela (notons le *SW*1), décrit son interface en WSDL et le publie au niveau d'un annuaire UDDI par exemple. Si une organisation *A* est intéressée par ce service, elle négocie un contrat d'utilisation avec *B*. Le contrat doit indiquer précisément les fonctions du service Web et ses caractéristiques (y compris les performances et la qualité de service attendues, la responsabilité de chaque partie, le paiement, des pénalités en cas d'abus, etc.), mais aussi des règles de sécurité relatives à l'invocation et à la fourniture des résultats du SW, l'ensemble de ces règles de sécurité formant la "politique-contrat" qui régit les interactions entre les organisations cliente et prestataire.

Pour simplifier, disons que *A* et *B* se mettent d'accord pour que "*les utilisateurs de A aient la permission de consulter les mesures de B en cas d'urgence*". Cet accord implique des mises à jours au niveau des politiques de *A* et *B*.

Au niveau de la politique locale de *A* :

- créer un objet "*Image de SW*1" et l'associer à une vue (e.g., *VA*) liée aux mesures.
- Associer une action "*invoke*()" sur "*Image de SW*1" a l'activité "*consultation*".
- Créer un rôle (notons le *RA*) qui aura le droit de consulter "*Image de SW*1" et lui associer des utilisateurs locaux (e.g., *Alice*). Il s'agit bien évidement des utilisateurs que *A* ait décidé (en interne) de laisser accéder au service de consultation des mesures de *B*. *B* n'a donc pas connaissance des rôles (internes à *A*) de ces utilisateurs, tout ce qu'elle a à savoir c'est qu'ils sont autorisés par *B* selon le contrat co-signé par *A* et *B*. *A*, par contre, assume ses responsabilités d'authentifier ses utilisateurs et de les associer à *RA*.

La règle OrBAC du côté de l'organisation cliente (*A*) est donc (Tableau 5) :

Tableau 5 : Dérivation des permissions au niveau de l'organisation cliente du Service Web.

Permission (*A*, *RA*, *VA*, *consultation*, *CA*) ∧ *Habilite* (*A*, *Alice*, *RA*) ∧ *Considère* (*A*, *invoke*(), *consultation*) ∧ *Utilise* (*A*, *image _SW*1, *VA*) ∧ *Définit* (*A*, *Alice*, *invoke*(), i*mage _SW1*, *CA*) → *Est_permis* (*Alice*, *invoke*(), *mage _SW*1)

De plus, il faut que l'action "*invoke*()" sur "*Image_SW*1" exécute un programme qui envoie le message SOAP d'appel du (vrai) service Web *SW*1 à l'organisation *B* (les types de paramètres et l'adresse du destinataire sont définis dans l'entrée UDDI correspondant à *SW*1, avec éventuellement des contraintes supplémentaires définies dans le contrat signé entre *A* et *B* pour *SW*1).

De l'autre côte de la communication, pour que l'exécution correspondant à cet appel de *SW*1 soit contrôlée dans l'organisation *B*, il faut que l'administrateur de sécurité de *B* ait créé un utilisateur virtuel "UV_A" qui joue un rôle (disons *RB* par exemple) lui permettant d'exécuter l'activité (*consultation*) correspondant au service Web *SW*1 et donc, d'accéder aux ressources de *B* rendues disponibles à certains utilisateurs de *A* (en l'occurrence ceux pouvant jouer *RA*). La règle dans la politique locale de *B* qui donne cette permission est la suivante :

Tableau 6 : Dérivation des permissions au niveau de l'organisation proposant le Service Web.

Permission (*B*, *RB*, *VB*, *consultation*, *urgence*) ∧ *Habilite* (*B*, UV_A, *RB*) ∧ *Considère* (*B*, *execute*(), *consultation*) ∧ *Utilise* (B, *ressource_SW*1, *VB*) ∧ *Définit* (*B*, UV_A, *execute*(), *ressource_SW1*, *urgence*) → *Est_Permis*(UV_A, *execute*(), *ressource_SW*1)

Notons qu'il faut aussi que la réception de l'appel par l'organisation *A* au service Web *SW*1 déclenche l'exécution par UV_A du programme réalisant toutes les actions requises pour *SW*1.

Par cet exemple, on voit que l'extension d'OrBAC pour les services Web ne concerne que la création dans l'organisation cliente d'objets correspondant à des images de services Web fournis par d'autres organisations, et de l'autre côté, la création dans l'organisation prestataire d'utilisateurs virtuels correspondant aux organisations clientes. L'image de service Web et l'utilisateur virtuel sont créés lors de la signature du contrat entre l'organisation cliente et l'organisation prestataire. Les entités de base d'OrBAC (organisation, sujet, action objet, rôle, activité, vue, contexte) ne sont pas changées, les règles de la politique locale concernant les services Web s'expriment donc de la même façon que les autres règles, et les mécanismes de sécurité qui les mettent en œuvre restent les mêmes.

b. Vérification des interactions dans PolyOrBAC

Nous avons montré que le modèle OrBAC permet bien d'exprimer les politiques de sécurité locales à chaque organisation participant à une IIC, même pour ce qui concerne les interactions avec

d’autres organisations. Cependant, cela ne permet pas de garantir que toutes les interactions sont légitimes : les différentes organisations étant mutuellement méfiantes, on peut supposer que l’organisation cliente peut avoir intérêt à abuser des services fournis par l’organisation prestataire, ou que cette dernière peut avoir intérêt à ne pas fournir le service prévu avec la qualité escomptée. Ceci est d'autant plus plausible que d'un côté, l'organisation cliente est la seule responsable de ses utilisateurs, et donc tenue responsable des actions qu'ils effectuent (dans notre exemple, l'authentification d'*Alice* ne se fait que par, et ne relève que de la responsabilité de *A*); de l'autre côté, l'organisation offrant le Service Web est responsable de ses services et des engagements qui vont avec, par exemple, la qualité de service attendue. Pour que notre cadriciel soit complet, il faut donc être capable de spécifier, mettre en œuvre et vérifier le respect des politiques-contrats établies entre les organisations. Ceci implique la capacité de réaliser les actions suivantes :

1. auditer les différents actions et échanges.
2. vérifier en temps réel à l’exécution si chaque interaction est légitime. C’est-à-dire si elle obéit aux règles exprimées dans la politique-contrat correspondante, et dans le cas contraire,
3. désigner l’organisation qui doit être tenue responsable de l’abus ou de la fraude. En effet, dans les systèmes complexes, l'expérience montre que même avec une politique de sécurité, des abus et violations peuvent se produire lors de l'exécution du système. Par exemple, même si une certaine autorisation a le droit d'invoquer un service, la multiplication des invocations dans un certain intervalle de temps pourrait conduire à un déni de service, ce qui n'est pas légitime. Dans le même sens, même si certaines actions sont légitimes, le processus qui les associe ainsi que certaines subtilités de leurs réalisations peuvent conduire à des canaux cachés, a priori non légitimes. Il serait donc souhaitable d'avoir un mécanisme qui détecte, en temps réel et de manière simple, ce type de comportements anormaux et notifie les parties concernées.

Pour cela, nous proposons de spécifier la politique-contrat avec des permissions, interdictions, et obligations portant sur les interactions liées aux services Web, c’est-à-dire sur des échanges de messages. Il s’agit donc plutôt de représenter des protocoles de communication par messages, et d'en surveiller l'exécution.

Nous avons remarqué que dans la plupart des cas, surveiller les politiques-contrats d’un service Web consiste en fait à surveiller les permissions, obligations, interdictions ainsi que les contraintes temporelles et provisionnelles (liées au *workflow* par exemple) sur les échanges de messages. Bien évidement, il est toujours possible de représenter ces besoins avec OrBAC et d'effectuer des vérifications hors-ligne (ce qui n'est d'ailleurs pas facile dans le cas général). Néanmoins, réaliser des vérifications en temps réel à l'exécution retarderait certainement le système, voir même le rendrait inutilisable, surtout dans le cas de systèmes complexes, distribués, ayant différentes politiques locales et de collaboration. Pour pallier ce type de problèmes pratiques, nous nous sommes tournés vers les automates temporisés [Alur & Dill, 1994], plus adaptés à des vérifications en-ligne notamment grâce aux techniques de vérification par modèle dites de *model-checking*.

La question qui se pose maintenant est comment représenter les politiques-contrats avec les automates temporisés. Ceci reviendrait à pouvoir exprimer les modalités d'accès (permissions, obligations, interdictions), les évènements et chemins redoutés, les processus à appliquer en cas de conflit, ainsi que les contraintes temporelles et provisionnelles. Pour les deux derniers points, il est évident que les automates temporisés sont bien adaptés. En effet, par définition même, un automate temporisé est un automate fini incluant un ensemble d'horloges (*clocks)*. Les transitions d’un automate

temporisé sont étiquetées par des gardes (*guards*), c'est-à-dire des conditions sur les valeurs d'horloges, des actions et des mises à jour, qui attribuent de nouvelles valeurs aux horloges. Reste maintenant à pouvoir exprimer les modalités d'accès et les conflits éventuels.

Une *permission* (correspondant à une action qui est autorisée par les clauses de la politique-contrat) est simplement représentée par le biais de transitions dans les automates temporisés. Par exemple, dans la figure 2, le système peut exécuter l'action *a* à tout moment, puis se comporter selon les possibilités définies dans l'automate *A*.

Pour les *interdictions*, nous en distinguons deux types dans les contrats :

- L'*interdiction implicite* : comme dans les politiques de contrôle d'accès positives, tout ce qui n'est pas explicitement autorisé est interdit, une action qui ne correspond à aucune transition dans l'automate est tout simplement considérée comme interdite.
- L'*interdiction explicite*: dans notre modèle, une interdiction explicite est représentée par une transition vers un *état d'échec* (illustré par une *frimousse* triste dans Figure 3). Les interdictions explicites, présentes dans OrBAC, permettent souvent d'exprimer des règles plus claires qu'une simple absence de permission. En particulier, cela permet d'exprimer des exceptions à des règles de permissions, ou de limiter la propagation des permissions dans les hiérarchies de rôles, ce qui peut être intéressant dans le cas des systèmes collaboratifs avec plusieurs politiques de sécurité définies par différents administrateurs. Dans notre modèle, la transition vers un état d'échec correspond à la détection d'une action interdite (supposée malveillante), pour laquelle on peut définir un traitement d'exception destiné à contrer l'action malveillante ou à corriger les dégâts qu'elle pourrait avoir causés. Le non-respect d'une interdiction pourra aussi conduire à imposer les pénalités à l'organisation responsable de l'action interdite.

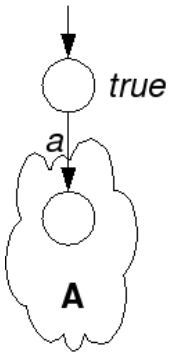


Figure 2 : Modélisation des permissions.

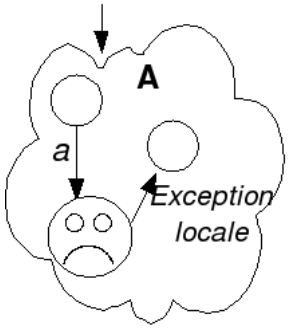


Figure 3 : Modélisation des interdictions implicites.

En ce qui concerne les obligations, nous les représentons par des transitions auxquelles nous associons des échéances temporelles (*time-outs*) sur des transitions et des invariants d'états : si la transition n'est pas activée avant l'échéance, une autre transition est déclenchée automatiquement vers un état d'échec, qui peut conduire à un traitement d'exception. Dans cette sémantique, l'obligation correspond à une action qui doit (obligatoirement) être réalisée dans un temps donné. Ceci est schématisé dans la figure 4, où *k* est un compteur de temps, *d* est l'échéance temporelle, *b* est l'action obligatoire, *a* est l'action déclenchée par l'échéance, conduisant au traitement d'exception *A*2. L'état à partir duquel l'action *b* est obligatoire possède un invariant ($k <= d$).

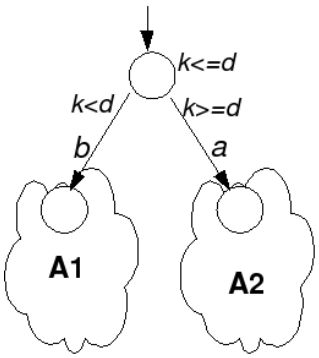


Figure 4 : Modélisation des obligations.

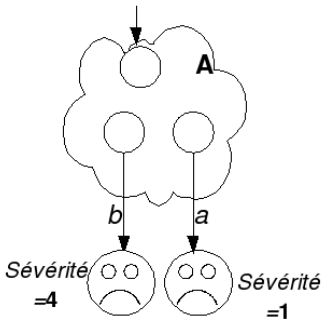


Figure 5 : Modélisation des situations de conflit.

Comme on l'a déjà précisé, quand une interdiction explicite est violée ou lorsqu'une obligation n'est pas remplie, l'automate atteint un état d'échec. Cette situation ne survient que si l'une des deux parties, le client ou le fournisseur du service Web, ne respecte pas les règles de la politique-contrat. Par l'analyse de la séquence états-transitions qui a conduit à l'état d'échec; la représentation par automate temporisé aide naturellement à identifier la partie responsable de cette violation de politique ainsi que le chemin qui a conduit à cela.

Cette modélisation des différents (situations de conflit) permet donc non seulement d'identifier les anomalies et les abus (violations de droits), mais aussi d'identifier les activités (succession d'actions et d'interactions) qui ont conduit à ces situations, d'en déduire le responsable de l'anomalie (celui qui a réalisé une action explicitement interdite, ou n'a pas réalisé une action obligatoire dans les temps prévus), et donc de lui imposer les pénalités prévues au contrat. De plus, comme les conséquences peuvent avoir des degrés différents de gravité et conduire à des pénalités graduées, ceci peut être représenté dans l'automate par des étiquettes (e.g., des degrés de sévérité) sur les états d'échec.

Par ailleurs, la modélisation des politiques-contrats par des automates temporisés permet de vérifier certaines propriétés par analyse statique (hors-ligne) telles que la sûreté[2], la vivacité[3], l'atteignabilité[4], l'absence de blocage[5] et l'équité[6]. En particulier, la cohérence de la politique correspond au fait que tous les états normaux (c'est-à-dire les états qui ne sont pas des états d'échec), doivent être accessibles ; autrement dit, le protocole est correct et correspond au fonctionnement normal. L'analyse statique permet également d'identifier tous les scénarios (séquences d'états-transitions) qui conduiraient aux états d'échec (événements redoutés), et de vérifier qu'ils correspondent bien à des violations de la politique-contrat.

Mais le principal intérêt de cette modélisation est de permettre une vérification en temps-réel à l'exécution, par ce qu'on appelle *run-time model checking*. Dans le cadriciel PolyOrBAC, ceci correspond à vérifier sur chaque échange de messages de service Web (seul moyen d'interaction entre organisations), si le message correspond à une transition autorisée dans le modèle de la politique-contrat, à bloquer tout message qui ne correspond pas à une transition autorisée (interdiction implicite), et enfin à déclencher des actions de recouvrement (traitement d'exception) si un message correspond à

[2] La propriété de sûreté (Safety) exprime que sous certaines conditions, un évènement ne peut jamais se produire.
[3] La Propriété de vivacité (Liveness) exprime que sous certaines conditions un évènement finira par avoir lieu
[4] La propriété d'atteignabilité (Reachability) indique qu'un état du système peut être atteint.
[5] La propriété d'absence de blocage (No deadlock) exprime que le système ne se trouvera jamais dans une situation où il ne peut plus progresser.
[6] La propriété d'équité faible (Fairness) exprime qu'un évènement arrivera (ou n'arrivera pas) infiniment souvent.

une action explicitement interdite, ou si on ne reçoit pas un message correspondant à une obligation avant l'échéance temporelle prévue. La vérification à l'exécution fait donc partie des mécanismes de sécurité mis en œuvre dans PolyOrBAC pour sécuriser les interactions entre organisations, en mettant en application les politiques-contrats.

Pour conclure cette partie, nous pouvons dire que PolyOrBAC permet de gérer, de manière sécurisée, les interactions entre des organisations qui collaborent dans un environnement de suspicion mutuelle, distribué, hétérogène et dynamique. Ces aspects résument en fait les besoins de sécurité qu'on trouvent notamment dans les infrastructures critiques. Le reste de cette section décrit brièvement comment PolyOrBAC peut s'appliquer à un scénario des infrastructures de production, de transport, et de distribution d'énergie électrique.

Application du cadriciel PolyOrBAC à une IC

Description du scénario de délestage semi-automatique

Dans la pratique, une ou plusieurs compagnies de production d'électricité sont responsables de plusieurs centrales électriques, connectées à un ou plusieurs réseaux de transport. Chaque réseau de transport se compose de lignes à très haute tension et de sous-stations de transport et est relié à des réseaux de distribution. Chaque réseau de distribution est composé de lignes à moyenne et basse tension et de sous-stations de distribution, et distribue de l'électricité aux abonnés (industries, secteur tertiaire, habitations, etc.). Un réseau de transport est géré par un centre de contrôle national (NCC pour *national control center*) et plusieurs centres de contrôle régionaux (RCC pour *regional control center*) tous contrôlés par un opérateur TSO (*transmission system operator*). Un réseau de distribution est géré par un centre de contrôle local (ACC pour *area control center*) contrôlé par un opérateur DSO (*distribution system operator*) qui peut agir sur des MCD-TU (pour *Monitoring and Control Data Terminal Unit*, qui peut être traduit par unité terminale de surveillance et de commande).

Appliquons maintenant PolyOrBAC à un scénario d'exécution particulier : le délestage semi-automatique de la distribution d'électricité dans certaines zones en cas de problèmes de sous-production, de surconsommation, ou encore de coupure de ligne de transport. En cas d'urgence, pour éviter un effet de cascade qui conduirait à un black-out, il faut réduire la distribution dans des zones non critiques. Ce scénario est donc particulièrement intéressant puisqu'il est susceptible d'être la cible d'attaquants désirant provoquer un black-out.

Comme indiqué dans la Figure 6, en fonctionnement normal, toutes les sous-stations de distribution (DS SS, pour *distribution system substations*) envoient des signaux et mesures (puissance, tension, fréquence) au centre de contrôle de distribution (DS ACC, pour *Distribution System Area Control Center*) où opère le DSO (Distribution System Operator). (1), qui lui-même transmet des signaux et mesures au centre de contrôle régional de transport (TS RCC, pour *Transmission System Regional Control Center*) où opère le TSO (3). De même, les sous-stations de transport (TS SS, pour *transmission system substation*) envoient différents signaux et mesures à leur TS RCC (2). Le TS RCC surveille le réseau électrique et son TSO (au vu des mesures et signaux reçus ou d'informations provenant du centre de contrôle national) et peut imaginer certaines situations d'urgence potentielles, qui pourraient nécessiter de réduire la puissance distribuée. Il prépare donc un plan de défense pour délester des zones de distribution particulières.

Pour mettre en œuvre ce plan de défense, le TSO envoie à chaque DSO des zones concernées des demandes de préparation de réduction de puissance (4). Le DSO en fonction de la réduction de puissance demandée et des conséquences relatives des coupures de courant dans les zones industrielles

ou d'habitation au moment du délestage éventuel, sélectionne des sous-stations DS SS et le DS ACC leur envoie des ordres d'armement (5). Ces DS SS arment alors leurs MCDTU et envoient un accusé de réception au DS ACC (6).

Lorsque suffisamment de DS SS sont armées pour pouvoir délester le réseau de distribution de la puissance demandée, le DS ACC envoie un accusé d'armement au TS RCC (7). Le TS RCC envoie alors un signal à la sous-station de transport TS SS de ce réseau de distribution (8) pour préparer un délestage automatique éventuel. Pendant ce temps, s'il le juge utile, le DSO peut armer des DS SS et en désarmer d'autres, à condition de maintenir la puissance "délestable" à un niveau supérieur à celui de la demande faite par le TSO.

En cas de détection d'une véritable situation d'urgence (déséquilibre entre la puissance disponible et la puissance consommée qui se traduit par une réduction de fréquence), la TS SS diffuse automatiquement un ordre de délestage (8) à toutes les DS SS de son réseau de distribution, et seules les DS SS armées précédemment déclenchent le délestage sur leurs MCDTUs (7). En revanche, si les conditions de risque disparaissent avant qu'une urgence survienne, le TSO peut envoyer une annulation de la demande de réduction de puissance aux DSO, qui eux-mêmes envoient un ordre de désarmement à toutes les DS SS armées (Figure 6).

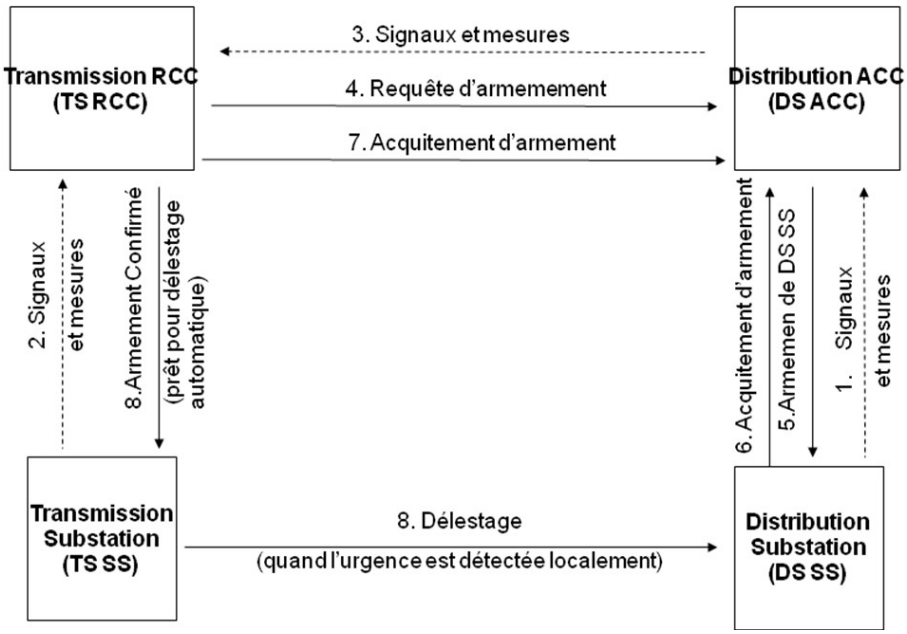


Figure 6 : Signaux et mesures échangés dans le scénario de délestage.

Tableau 7 distingue les différents services Web impliqués dans le scénario de délestage, en déterminant le client et le fournisseur de chaque service. Dans ce scénario, nous identifions quatre organisations (TS RCC, DS ACC, TS SS, DS SS) et cinq services (SW1-demande_d'armement, SW2-ordre_d'armement, SW3-préparation_de_délestage, SW4- activation_de_délestage, SW5-réintégration).

Tableau 7 : Services Web utilisés dans le scénario.

Service	Prestataire	Client
SW1-demande_d'armement	DS ACC	Opérateur TSO
SW2-ordre_d'armement	DS SS	Opérateur DSO
SW3-préparation_de_délestage	TS SS	EMS[7] au niveau du TS RCC
SW4- activation_de_délestage	DS SS	Processus sentinelle de la TS SS
SW5-réintégration	DS SS	Opérateur DSO

La répartition de ces SW ainsi que les différentes invocations sont décrites dans la Figure 8.

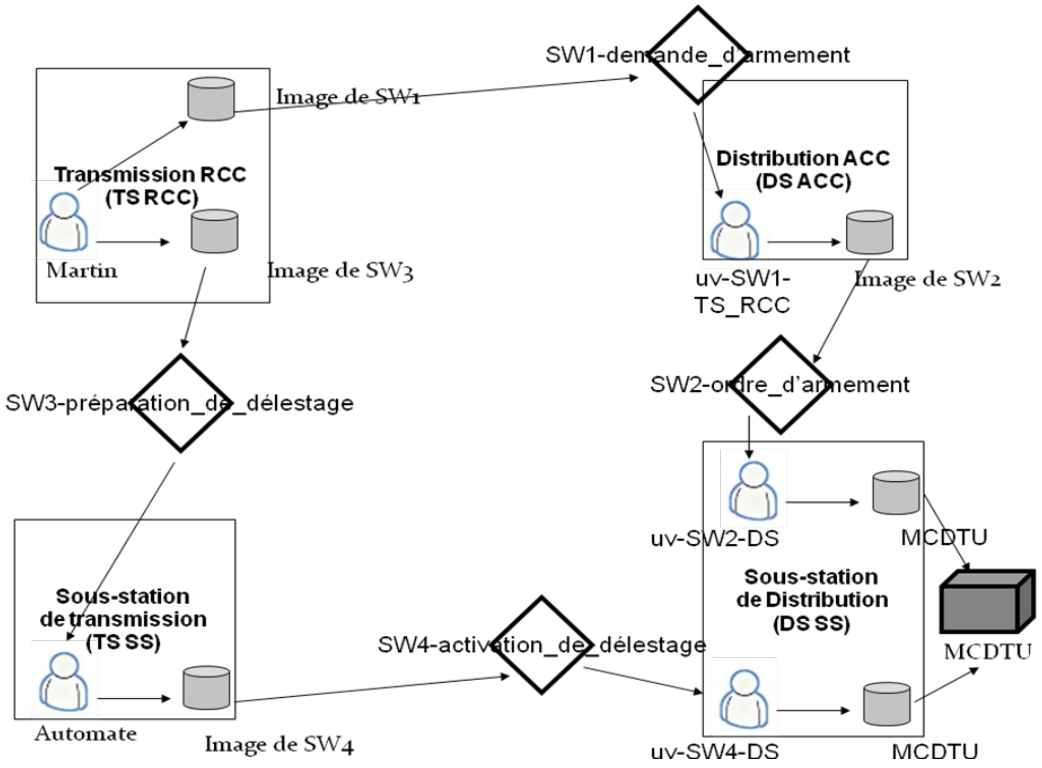


Figure 7 : Application des notions d'image de service Web et d'utilisateur virtuel au scénario.

Dans ce qui suit, nous détaillons l'utilisation de *SW*1-*demande_d'armement*. Pour les autres Services, un descriptif complet est disponible dans [Abou El Kalam *et al.* 2009]. *SW*1-*demande_d'armement* gère les interactions entre le TS RCC et le DS ACC. Ces interactions correspondent à la demande d'armement par le TSO (donc l'organisation TS RCC) au DSO (donc l'organisation DS ACC), ainsi qu'à la demande de désarmement (si la pré-urgence disparaît), et à l'ordre de redémarrage (une fois que l'urgence a disparu).

Règles OrBAC pour la demande d'armement par le TS RCC

Dans un premier temps, l'opérateur TSO constate un état de pré-urgence et envoie la requête de demande d'armement (*SW*1-*demande_d'armement*) à la DS ACC. Lorsque la personne (par exemple,

[7] Energy Management System

Martin) qui est l'opérateur TSO invoque le service *SW*1, cela correspond à l'exécution de l'action *invoquer*() sur l'objet *image_SW1*. Cette action est permise par la politique OrBAC du TS CC si l'utilisateur Martin a été authentifié, s'il joue bien le rôle TSO, et s'il existe une règle de permission pour le rôle TSO de réaliser l'activité correspondant à cette action sur la vue correspondant à cet objet dans le contexte de pré-urgence actuel. La séquence de règles OrBAC suivante présente les droits d'accès pour l'utilisation du service SW1 au niveau de l'organisation TS RCC.

Tableau 8 : Représentation de la règle OrBAC pour SW1 du côté TS RCC.

Permission(*TS RCC*, *TSO*, *accéder*, *vue_SW*1, *pré-urgence*) ∧ *Habilite*(*TS RCC*, *Martin*, *TSO*) ∧ *Considère*(*TS RCC*, *invoquer*(), *accéder*) ∧ *Utilise*(*TS RCC* C, *image_SW*1, *vue_SW*1) ∧ *Définit*(*TS RCC*, Martin, *invoquer*(), *image_SW*1, pré-urgence) *Est_permis*(*Martin*, *invoquer*(), *image_SW*1)

Cette séquence peut être interprétée comme suit : la politique OrBAC du TS RCC contient un certain nombre de règles pour gérer le service Web SW1-demande_d'armement. Première règle : au niveau de la TS RCC, le rôle TSO a le droit, dans un contexte de pré-urgence, d'exécuter les actions incluses dans l'activité *accéder* sur la vue *vue_SW1* correspondant à SW1. Deuxième règle : nous attribuons le rôle TSO à Martin. Troisième règle : nous incluons l'action *invoquer*() dans l'activité *accéder*. Quatrième règle : nous incluons l'objet *image_SW1* dans la vue *vue_SW1*. Cinquième règle : nous précisions que le sujet Martin a la possibilité de réaliser l'action *accéder* sur l'objet *image_SW1* dans un contexte spécifique pré-urgence. De cet ensemble de règles on déduit le prédicat Est_permis qui affirme que le sujet Martin a le droit d'exécuter l'action *accéder* sur l'objet *image_SW1*.

Par ailleurs, pour chaque Service Web un contrat doit être signé par le client et le prestataire, contenant la politique-contrat qui contrôle les échanges de messages. Cette politique-contrat est définie par deux automates temporisés, l'un du côté client, l'autre du côté prestataire, chacun installé dans le CIS de son organisation. La section suivante présente l'automate temporisé correspondant à *SW*1-*demande_d'armement* du côté du TS RCC.

Automate de *SW*1-*demande_d'armement* au niveau du TS RCC

Du côté du TS RCC (le client de SW1), selon la politique-contrat du service SW1 et comme illustré dans Figure 8, l'automate attend une invocation pour une demande d'armement (en provenance du TSO). Quand le message correspondant à cette invocation est intercepté par le TS RCC, la transition correspondante (*WS1-arming-request*) est activée dans l'automate, une minuterie (*timer*) est initialisée (à 10 mn dans l'exemple de la figure) et l'automate atteint un état où il attend un acquittement *WS1-arming-request-ack* du DS ACC. Si le délai expire sans avoir reçu l'accusé de réception de la part du DS ACC, un message *WS1-arming-request-error* est envoyé au TS RCC pour déclencher un traitement d'exception correspondant à cette situation, et l'automate atteint l'état d'échec *arming_failure*. Inversement, dans des situations normales, lorsque le TS RCC reçoit l'acquittement *WS1-arming-request-ack*, son automate va vers l'état où il sera prêt pour une action d'urgence, alors que la réception du message par le TS RCC déclenche automatiquement le processus EMS, qui lui même invoquera le service SW3.

Dans cet état, si la situation de pré-urgence disparaît, le TSO peut décider de désarmer l'ensemble des sous-stations qui ont été armées. Pour cela, le TS RCC envoie le message *WS1-*

disarming-request vers le DS ACC et attend l'acquittement *WS1-disarming-request-ack* du DS ACC, qui remettra l'automate dans l'état initial. Si le délai (fixé à 10 unités de temps dans la figure 43) expire sans avoir reçu l'acquittement *WS1-disarming-request-ack* du DS ACC, un traitement d'exception similaire au précédent est déclenché.

Au contraire, si la situation d'urgence survient et que le TS SS déclenche le délestage, le TSO doit surveiller la fin de l'urgence, c'est-à-dire le moment où il peut demander le redémarrage du service au DS ACC, qui reconnectera les DS SS qui ont été délestés. Cette action se traduit par l'envoi du message *WS1-restatring-request*, puis l'attente de la confirmation *WS1-disarming-request-ack*, avant de revenir à l'état initial.

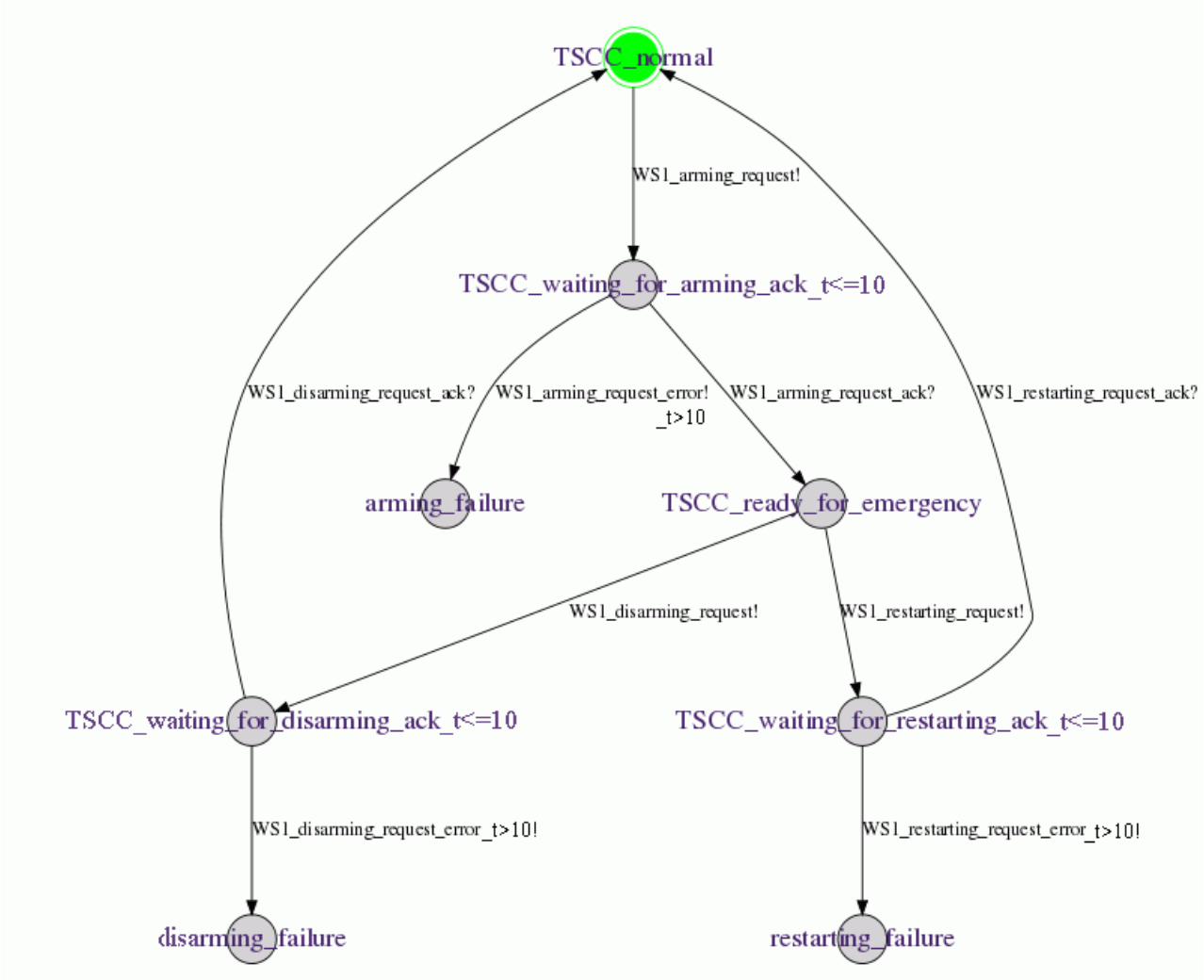


Figure 8 : Automate de SW1-demande_d'armement au niveau du TS RCC.

Règles OrBAC de SW1-demande_d'armement dans la politique du DS ACC

Du côté du DS ACC, la réception de la requête de SW1-demande_d'armement déclenche l'exécution pour le compte de l'utilisateur virtuel *uv-SW1-TS_RCC* d'un processus qui affiche un message urgent sur la console du DSO. Cet accès (*uv-SW1-TS_RCC* à la console) est vérifié selon la politique de contrôle d'accès du DS ACC et est accordée en fonction de la séquence OrBAC décrite dans Tableau 9.

Tableau 9 : Représentation de la règle OrBAC pour SW1 du côté DS ACC.

Permission(*DS ACC*, *Rôle_SW1*, *afficher*, *console_DSO*) ∧ *Habilite*(*DS ACC*, *UV-SW1-TS_RCC*, *Rôle_SW1*) ∧ *Considère*(*DS ACC*, *écrire*(), *afficher*) ∧ *Utilise*(*DS ACC*, *terminal*_1, *console_DSO*) ∧ *Définit*(*DS ACC*, *UV-SW1-TS_RCC*, *écrire*(), *terminal*_1) →*Est_permis*(*UV-SW1-TS_RCC*, *écrire*(), *terminal*_1)

Première règle : le rôle *Rôle_SW1* a le droit d'afficher des messages d'urgence (activité *afficher*) sur la console du DSO (vue *console_DSO*). Deuxième règle : nous attribuons le rôle *Rôle_SW1* au sujet UV-*SW1-TS_RCC,* qui représente le TS RCC pour exécuter le service *SW*1. Troisième règle : nous incluons l'action *écrire*() dans l'activité *afficher.* Quatrième règle : nous incluons l'objet *terminal_1* dans la vue *console_DSO*. Cinquième règle : nous précisions que le sujet UV-*SW1-TS_RCC* a la possibilité de réaliser l'action *écrire*() sur l'objet *terminal_1*, quelque soit le contexte. De cet ensemble de règles, on déduit le prédicat Est_permis qui affirme que le sujet UV-*SW1-TS_RCC*, a le droit d'exécuter l'action *écrire*() sur l'objet *terminal_1*.

L'automate de SW1-demande_d'armement dans le CIS du DS ACC

Nous allons maintenant analyser la façon dont l'automate de la politique-contrat liée à SW1-demande_d'armement vérifie dans le CIS du DS ACC vérifie les invocations en provenance du côté client (figure 44). Au départ, cet automate est en attente du message *WS1-arming-request* du TS RCC. Quand ce message est intercepté, une minuterie est initialisée (à la valeur de 10 mn dans l'exemple de la figure), et l'automate atteint un état où il attend la réalisation d'une obligation, l'armement de suffisamment de DS SS, qui se traduira par l'envoi d'un acquittement *WS1-arming-request-ack* par le DS ACC généré automatiquement par le processus lancé par le DSO pour armer les DS SS. Si ce message n'est pas intercepté par le CIS du DS ACC avant que la minuterie ne se déclenche, l'automate atteint un état d'échec (avec envoi du message *WS1-arming-request-error* au DS ACC pour déclencher un traitement d'exception.

Au contraire, si l'acquittement *WS1-arming-request-ack* est bien intercepté, l'automate atteint un état d'attente, où il reste :

- jusqu'à la réception d'un message *WS1-disarming-request* du TS RCC qui déclenchera le désarmement par le DSO des DS SS, puis le retour à l'état initial après réception de l'acquittement *WS1-disarming-request-ack*. Si ce message n'est pas reçu dans les temps prévus, un traitement d'exception est déclenché.
- ou la réception d'un message de réinitialisation *WS1-restarting-request* provenant du TS RCC (après la fin d'une urgence qui aura provoqué un délestage), qui demandera au DSO de redémarrer les DS SS qui ont déclenché le délestage. Une fois que ces DS SS auront redémarré (interception de l'acquittement *WS1-restarting-request_ack*), l'automate revient dans son état initial. Si ce message n'est pas reçu dans les temps prévus, un traitement d'exception est déclenché.

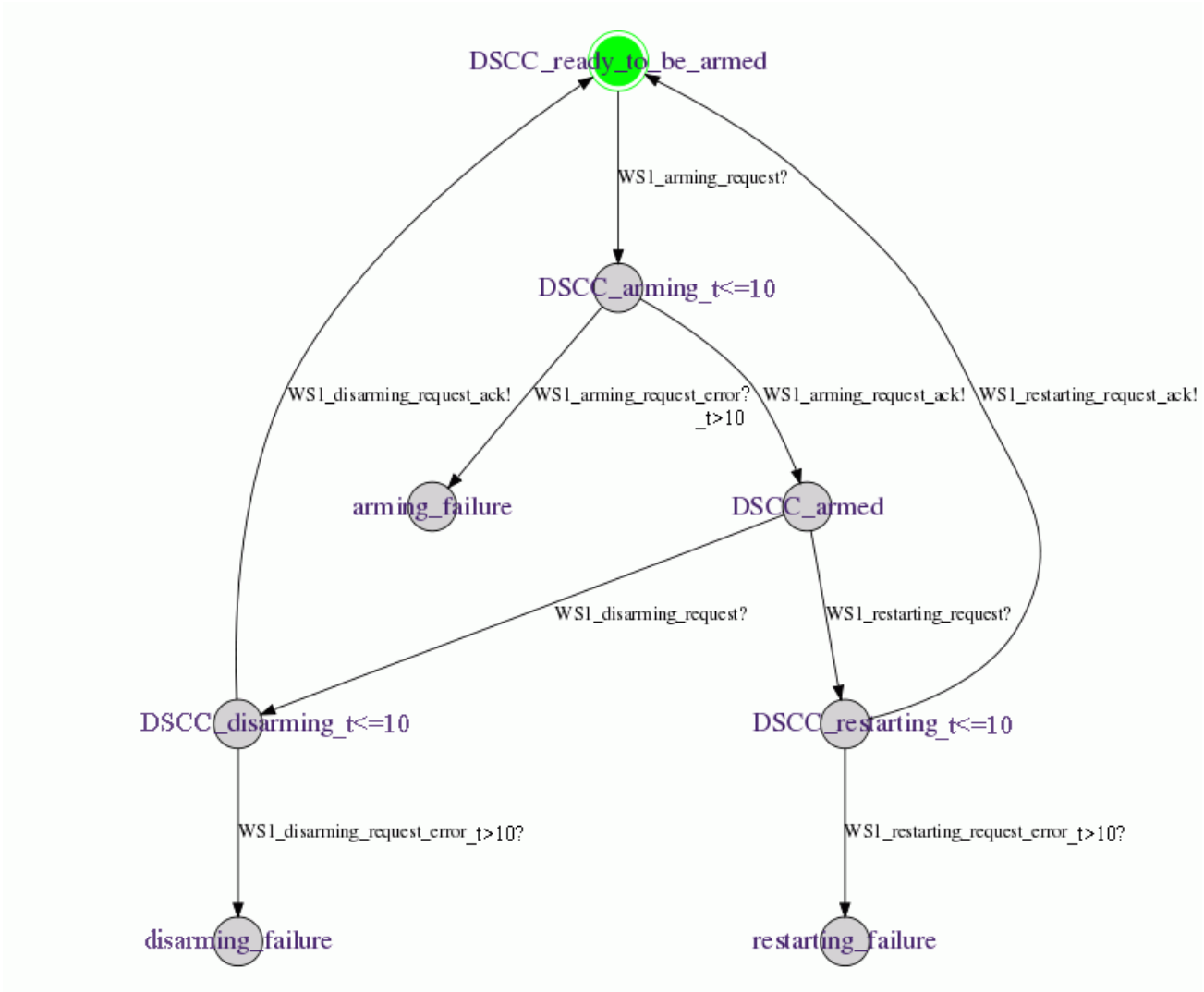


Figure 9 : Automate de SW1-demande_d'armement au niveau du DS ACC.

Enfin, notons qu'un prototype a été développé pour montrer la possibilité de mise en œuvre de la plate-forme PolyOrBAC au sein d'un environnement physique réel, tout en tenant compte des différentes contraintes des systèmes et réseaux, puis en démontrant l'implémentation de chaque composant (collaboration grâce à la technologie des services Web, contrôle d'accès grâce au modèle OrBAC, et vérification des interactions grâce aux politiques-contrats et à leur représentation par automates temporisés).

V. Les recommandations

A. *Motivation*

Malheureusement, alors que les politiques de sécurité jouent un rôle important dans le processus de sécurisation des réseaux et systèmes, la plupart des recherches, implémentations et outils sont limités aux permissions et interdictions. Plus récemment, des travaux se sont intéressés aux obligations, notamment Nomad, XACML, OrBAC et bien d'autres [Cuppens *et al.* 2005, Betini *et al.* 2002, Demeanor *et al.* 2001, Ni *et al.* 2008]. A ces trois modalités d'accès, OrBAC ajoute le concept de recommandation, sans pour autant expliquer son fonctionnement (syntaxe, sémantique et axiomatique). Convaincus de l'importance des recommandations dans l'expression des politiques de sécurité, et n'ayant trouvé aucun travail publié autour de ce concept, nous avons décidé d'en introduire les briques de base.

En effet, nous disposons d'une réglementation abondante autour des recommandations [Abou El Kalam 2008a, Abou El Kalam & Balbiani 2009]. Par exemple, dans le domaine qui nous intéresse (infrastructures critiques), plusieurs organisations comme le Conseil de l'Europe [EC 1994], le Conseil International de gestion des risques (IGCR, pour *International Risk Governance Council*) [IRGC 2007], le Conseil nord-américain de la fiabilité électrique (NERC, pour *North American Electric Reliability Council*) ont produit plusieurs recommandations pour la protection des infrastructures critiques, en particulier celle de l'électricité [NERC 2003]. De la même manière, une grande partie de la législation du domaine médical est en fait liée à des recommandations ou à des directives, par exemple, les recommandations de l'Assemblée Générale des Nations-Unis [Résolution 1990], les recommandations de Conseil de l'Europe [Recommandation 1992, [Recommandation 1997], les directives du parlement européen [Directive 1995]. Globalement, l'ensemble de ces documents liste une série de règles du type : "*il est recommandé de ...*", "*il est déconseillé de ...*".

Par exemple, la loi 2002-303 relative aux droits des patients [Loi 2002] donne au patient la possibilité d'accéder à son dossier médical ; toutefois, elle recommande la présence de son médecin traitant. En effet, celui-ci est plus apte à comprendre les notions marquées au dossier médical et de les présenter de manière pédagogique et constructive au patient. La même loi stipule que si le patient est mineur ou souffre de troubles psychologiques, la présence de son tuteur est fortement recommandée.

A partir de cet exemple, on peut remarquer que d'un point de vue conceptuel, une recommandation est une modalité d'accès qui est plus forte que les permissions (comme le patient assume ses conséquences s'il ne respecte pas cette recommandation), mais moins forte que les obligations (car il peut toujours ne pas respecter l'obligation et accéder à son dossier médical).

Prenons un autre exemple : les recommandations R (95) 5 et R (95) 4 du Conseil de l'Europe sur la protection des données à caractère personnel recommandent que les données personnelles et médicales soient obtenues du sujet lui-même [Recommandation 95, Recommandation 97]. C'est bien évidement moins fort qu'une obligation (étant donnée que les données médicales peuvent être demandées à d'autres sources, par exemple si le sujet n'est pas en situation de les fournir) ; mais en même temps, cet accès est plus fort que les permissions car le sujet peut demander des explications voire des justifications si cette recommandation n'est pas respectée.

Toujours dans le même sens, plusieurs organisations telles que les Centres de veille en sécurité

(CERT) ou le W3C (*World Wide Web Consortium*) ainsi que certains constructeurs comme CISCO publient régulièrement des recommandations qu'il serait fort souhaitable de respecter. D'ailleurs, dans ses standards et RFC, l'IETF associe le verbe "*Should*" à une recommandation. Il précise dans la RFC 2119 que "*must*", "*required*" et "*shall*" sont associés à une exigence absolue de la spécification ; "*must not*" et "*shall not*" sont associés à une interdiction ; "*should*" ainsi que l'adjectif "***recommended***" signifient qu'il peut y avoir des raisons valides dans certaines circonstances pour ignorer un certain élément, mais que l'implication globale doit être comprise et bien pesée avant de choisir autre chose ; "*should not*" et "***not recommended***" signifient qu'il peut y avoir des raisons valides dans certaines circonstances quand le comportement particulier est acceptable ou même utile, mais les implications globales doivent être comprises et bien pesées avant d'implémenter un comportement décrit avec cette étiquette [RFC 2119].

On peut donner de multiples autres exemples, mais globalement, on peut conclure qu'il y a un fort besoin de gérer ce concept de recommandation, de l'exprimer au sein de la politique de sécurité, de se doter d'un cadre logique qui permet de le formaliser et de raisonner dessus (e.g., vérifier les incohérences, interroger la politique), puis de développer les mécanismes nécessaires pour l'implémenter dans des systèmes réels.

Dans le reste de ce chapitre, nous présentons notre nouveau cadre logique pour modéliser les recommandations. En particulier, nous définissons notre langage de spécification des recommandations baptisé RSL, pour *Recommendation Specification Language*, et nous précisons sa sémantique, son axiomatique et ses conditions de vérité [Abou El Kalam 2008a, Abou El Kalam & Balbiani 2009].

B. *Langage de spécification des recommandations*

Syntaxe

Le choix d'un langage formel pour la spécification de la politique de sécurité est basé, d'une part sur les besoins de l'application, et d'autres part sur la richesse du langage. Pour spécifier une politique de sécurité, nous devons exprimer des normes, c'est-à-dire des règles qui précisent ce qui doit être fait et ce qui ne doit pas l'être. Ceci est possible avec des formalismes tels que la logique déontique ; celle-ci permet de représenter naturellement des notions comme les permissions, obligations et interdictions.

Plus précisément, la logique déontique [Aqvist 1966, Prior 1954, Bieber & F. Cuppens 1991, Glasgow *et al.* 1992] est une branche de la logique modale [Chella 1980 ; Catach 1989] qui assimile les connecteurs intentionnels de la *nécessité* ( ) et de la *possibilité* (◊) à des obligations et des permissions respectivement. Le langage de la logique déontique noté $L_O(\mathbf{\Phi})$ est l'ensemble des formules construit par les règles suivantes $f ::= p|\neg f \mid f \vee \phi \mid \mathbf{O}f$, où p est une proposition, f est une formule, $\mathbf{O}f$ signifie "*il est obligatoire que p*".

Les autres opérateurs modaux **F** (*il est interdit*), **P** (*il est permis*), **E** (*il est électif*) sont décrit par :

$$\mathbf{F}\phi = \mathbf{O}\neg\phi \quad (1)$$

$$\mathbf{P}\phi = \neg\mathbf{O}\neg\phi \quad (2)$$

$$\mathbf{E}\phi = \neg\mathbf{O}\phi \quad (3)$$

Une norme est caractérisée par la cohérence de ses obligations, ce qui correspond à la formule :

$$\neg(\mathbf{O}\phi \wedge \mathbf{O}\neg\phi) \quad (4)$$

Combinée avec $\mathbf{F}\phi=\mathbf{O}\neg\phi$, cette formule nous permet de déduire :

$$\neg(\mathbf{F}\phi\wedge\mathbf{F}\neg\phi) \quad (5)$$
$$\neg(\mathbf{O}\phi\wedge\mathbf{F}\neg\phi) \quad (6)$$

De plus, si on utilise les équivalences $\neg\mathbf{O}\neg\phi \leftrightarrow \mathbf{E}\neg\phi$ et $\neg\mathbf{F}\neg\phi \leftrightarrow \mathbf{P}\neg\phi$, on peut déduire que :

$$\mathbf{O}\phi\rightarrow\mathbf{E}\neg\phi \quad (7)$$
$$\mathbf{F}\phi\rightarrow\mathbf{P}\neg\phi \quad (8)$$

La combinaison des formules (2) (3) et (7) nous permettent de déduire que :

$$\mathbf{O}\phi\rightarrow\mathbf{P}\phi \quad (9)$$

Et de la même manière, en combinant (2) (3) et (8), on obtient :

$$\mathbf{F}\phi\rightarrow\mathbf{E}\phi \quad (10)$$

Par ailleurs, afin de tenir compte du concept de recommandation, nous avons étendu cette syntaxe de deux manières différentes [Abou El Kalam 2008a] et [Abou El Kalam & Balbiani 2009]. Par manque d'espace, nous nous limitons dans ce rapport à la description du travail présenté dans [Abou El Kalam & Balbiani 2009].

Nous avons tout d'abord étendu le langage de la logique déontique à l'ensemble des formules construites par les règles suivantes $f ::= p|\neg f \mid f \vee \phi \mid \mathbf{O}f \mid \mathbf{R}f$, **R** étant l'opérateur de recommandation. Par exemple, si la formule (*Lire*, *Bob*, *Guide d'utilisation*) exprime le fait que *Bob* lise le guide d'utilisation, la formule **R**(*Lire*, *Bob*, *Guide d'utilisation*) signifie qu'il est recommandé que *Bob* lise le guide d'utilisation.

De plus, afin d'exprimer des règles du type "*il est déconseillé que...*", nous introduisons l'opérateur modal "**I**" (pour *Inadvisable* en anglais). Et comme, déconseiller quelque chose revient à recommander de ne pas le faire, nous statuons que :

$$\mathbf{I}\phi=\mathbf{R}\neg\phi \quad (11)$$

Ainsi, la formule **R**(*Exécuter*, *Bob*, *Versions antérieures*) signifie qu'il est déconseillé d'exécuter les versions antérieures ; autrement dit, il est recommandé de ne pas exécuter les versions antérieures.

La cohérence de notre nouvel ensemble de formules correspond à la vérification de la formule :

$$\neg(\mathbf{R}\phi\wedge\mathbf{R}\neg\phi) \quad (12)$$

En tenant compte de (11), on trouve que la formule (12) est équivalente à :

$$\neg(\mathbf{I}\phi\wedge\mathbf{I}\neg\phi) \quad (13)$$
$$\neg(\mathbf{R}\phi\wedge\mathbf{I}\phi) \quad (14)$$

Ce qui revient à exclure les situations où quelque chose soit à la fois recommandée et déconseillée.

La question qui se pose maintenant est comment lier les obligations, recommandations et interdictions d'un côté ; et ce qui est interdit, déconseillé et "électif" d'un autre côté. La sémantique et l'axiomatique des deux sections suivantes nous permettra de déduire, entre autres, les formules suivantes :

$$\mathbf{O}\phi\rightarrow\mathbf{R}\phi \quad (15)$$

$\mathbf{R}\phi \rightarrow \mathbf{P}\phi$ (16)

$\mathbf{F}\phi \rightarrow \mathbf{I}\phi$ (17)

$\mathbf{I}\phi \rightarrow \mathbf{E}\phi$ (18)

Sémantique

La *sémantique* associée à une logique modale normale est appelée sémantique de *Kripke*, ou encore sémantique des mondes possibles [Kripke 1963]. Un modèle de *Kripke* $\boldsymbol{M}$ est un triplet ($\boldsymbol{W}$, $\mathfrak{R}$, $\boldsymbol{V}$) où

- $\boldsymbol{W}$ est un ensemble de mondes possibles *w*,
- $\mathfrak{R}$ est une relation binaire sur *W* appelée relation d'accessibilité, et
- $\boldsymbol{V}$: W×Φ → {vrai, faux} est une fonction qui donne pour chaque monde $w \in W$ la valeur de vérité *V*(*w*,*p*) de la proposition atomique *p*. La fonction *V* peut être étendue en $\overline{V}$ définie comme suit :
- $p \in \overline{V}(x)$ si et seulement si $p \in V(x)$;
- $\neg\phi \in \overline{V}(x)$ si et seulement si $\phi \notin \overline{V}(x)$;
- $\phi \vee \psi \in \overline{V}(x)$ si et seulement si $\phi \in \overline{V}(x)$ ou $\psi \in \overline{V}(x)$
- $\mathbf{O}\phi \in \overline{V}(x)$ si et seulement si pour tous les états *y* tel que $x\mathfrak{R}y$, $\phi \in \overline{V}(y)$.

Si on considère les relations (1), (2) et (3), on obtient :

- $\mathbf{F}\phi \in \overline{V}(x)$ si et seulement si pour tous les états *y* tel que $x\mathfrak{R}y$, $\phi \notin \overline{V}(y)$
- $\mathbf{P}\phi \in \overline{V}(x)$ si et seulement si pour certains états *y* tel que $x\mathfrak{R}y$, $\phi \in \overline{V}(y)$
- $\mathbf{E}\phi \in \overline{V}(x)$ si et seulement si pour certains états *y* tel que $x\mathfrak{R}y$, $\phi \notin \overline{V}(y)$

Dans Figure 10, à l'état *x*, *q* et *s* sont permis (présents dans certains états accessibles), tandis que *p* est obligatoire (présent dans tous les états accessibles).

Définissons maintenant les notions de "*satisfiabilité*" et "*validité*" dans notre modèle :

- Une formule ϕ est *valide* dans notre modèle $M=(W, \mathfrak{R}, V)$ si est seulement si $\phi \in V(x)$ pour tous les états *x*
- Une formule ϕ est "*satisfiable*" dans $M=(W, \mathfrak{R}, V)$ si est seulement si $\neg\phi$ est non valide dans *M.*

Notons que ces notions de *satisfiabilité* et validité proviennent de la sémantique de la logique modale. La théorie des correspondances dans la logique modale stipule que la validité des formules modales $\neg(\mathbf{O}\phi \wedge \mathbf{O}\neg\phi)$, $\neg(\mathbf{F}\phi \wedge \mathbf{F}\neg\phi)$, et $\neg(\mathbf{O}\phi \wedge \mathbf{F}\neg\phi)$ définies précédemment (3, 4 et 5) est reliée à la condition de sérialité. Une relation $\mathfrak{R}$ est dite sérielle si pour tous les états *x*, s'il existe au moins un état *y* accessible à partir de *x*, c'est-à-dire $x\mathfrak{R}y$.

Dans la suite, nous considérons que la relation $\mathfrak{R}$ de notre modèle est sérielle. Avec cette hypothèse, on peut facilement vérifier que :

- $\mathbf{O}\phi \in \overline{V}(x)$ si et seulement si $\mathfrak{R}(x) \cap \{y: \phi \in \overline{V}(y)\} = \mathfrak{R}(x)$, i.e., $\{y: \phi \in \overline{V}(y)\}$ *couvre entièrement* $\mathfrak{R}(x)$,
- $\mathbf{P}\phi \in \overline{V}(x)$ si et seulement si $\mathfrak{R}(x) \cap \{y: \phi \in \overline{V}(y)\} \neq \varnothing$ (l'ensemble vide), c'est-à-dire $\{y: \phi \in \overline{V}(y)\}$ *couvre partiellement* $\mathfrak{R}(x)$.

Intuitivement, on avait considéré qu'une recommandation est moins forte qu'une obligation, mais plus forte qu'une permission. Autrement dit, si on souhaite que les formules $\mathbf{O}\phi \rightarrow \mathbf{R}\phi$ et $\mathbf{R}\phi \rightarrow \mathbf{P}\phi$ soient valides, l'interprétation de l'opérateur modale de recommandation doit être entre les notions de "*couvre totalement*" (qui est associé aux obligations) et "*couvre partiellement*" (qui est associé aux permissions). Nous définissons ainsi la recommandation de la manière suivante :

- $\mathbf{R}\phi \in \overline{V}(x)$ si et seulement si $\{y: \phi \in \overline{V}(y)\}$ *couvre une grande partie de* $\Re(x)$.

De la même manière, comme nous avons défini ce qui est déconseillé à mi-chemin entre ce qui est interdit et ce qui est "*électif*", nous pouvons déduire que :

- $\mathbf{I}\phi \in \overline{V}(x)$ si est seulement si $\{y: \phi \in \overline{V}(y)\}$ *couvre une petite partie de* $\Re(x)$.

Ainsi, en introduisant la notion de recommandation, notre modèle sera définit par le quadruplet ($\boldsymbol{W}$, $\boldsymbol{\Re}$, $\boldsymbol{N}$, $\boldsymbol{V}$), où $\boldsymbol{N}$ est une fonction de voisinage qui associe, à chaque état x dans W, un ensemble $N(x)$ de sous-ensembles de $\Re(x)$. Pour tout état x, $N(x)$ peut être vue comme l'ensemble des sous états larges de $\Re(x)$. Ce sous-ensemble large caractérise ainsi l'ensemble des recommandations dans x.

À cet égard, la fonction de valuation V peut être étendue à la fonction V comme suit.

- $\mathbf{R}\phi \in \overline{V}(x)$ si est seulement si $\Re(x) \cap \{y: \phi \in \overline{V}(y)\} \in N(x)$

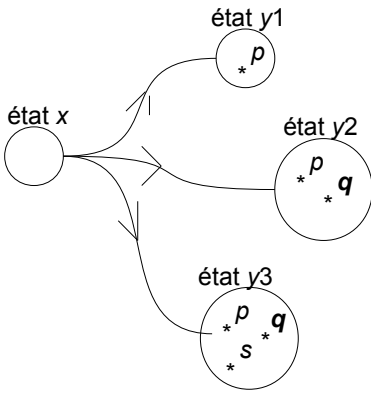


Fig. 10 : Exemple de permissions et d'obligations.

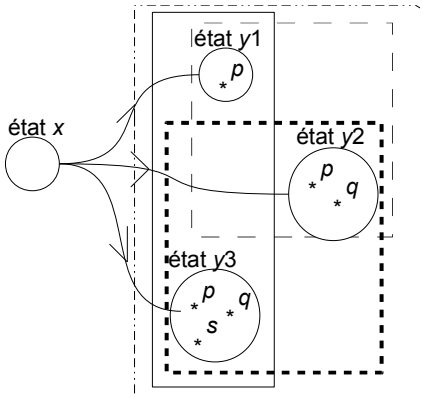


Fig. 11 : Exemple de sous-ensemble large.

Par exemple, si on considère le modèle de la figure 11 (obtenu à partir de la figure 10), $\Re(x) = \{\{y1,y2\},\{y2,y3\},\{y1,y3\},\{y1,y2,y3\}\}$. Et comme $\{y2,y3\}$ est considéré comme un sous-ensemble large de $\Re(x)$, $\{y2,y3\} \in N(x)$. Ainsi, q est recommandé à l'état x alors que s ne l'est pas.

Axiomatique

La définition d'un système logique nécessite de spécifier sa syntaxe, sa sémantique (présentées ci-dessous) mais aussi son axiomatique et ses règles d'inférence. Dans ce sens, notre langage considère, en plus des axiomes classiques de la logique propositionnelle, les axiomes suivants :

- $\mathbf{O}(\phi \rightarrow \psi) \rightarrow (\mathbf{O}\phi \rightarrow \mathbf{O}\psi)$ (19)
- $\mathbf{O}\phi \rightarrow \mathbf{P}\phi$ (20)
- $\mathbf{O}(\phi \leftrightarrow \psi) \rightarrow (\mathbf{R}\phi \leftrightarrow \mathbf{R}\psi)$ (21)
- $\mathbf{O}\phi \rightarrow \mathbf{R}\phi$ (22)
- $\mathbf{R}\phi \rightarrow \mathbf{P}\phi$ (23)
- $\mathbf{F}\phi \rightarrow \mathbf{I}\phi$ (24)
- $\mathbf{I}\phi \rightarrow \mathbf{E}\phi$ (24)

L'axiome (19) est connu sous l'axiome (*K*). Concrètement, il correspond au fait que l'opérateur modal d'obligation est interprété par une relation binaire. L'axiome (20), connu sous l'axiome (D), correspond au fait que la relation d'accessibilité $\mathfrak{R}$ est sérielle.

Et si on considère que $\mathfrak{R}(x) \in N(x)$ et que $\varnothing$[8] $\notin N(x)$, on peut facilement déduire la validité des formules $\mathbf{O}\phi \rightarrow \mathbf{R}\phi$ et $\mathbf{R}\phi \rightarrow \mathbf{P}\phi$. Toujours sous cette condition, et en considérant les formules (1), (3) et (11), on peut déduire que $\mathbf{F}\phi \rightarrow \mathbf{I}\phi$ et $\mathbf{I}\phi \rightarrow \mathbf{E}\phi$ sont également valides.

Par ailleurs, en plus des règles d'inférence classiques de la logique propositionnelle, nous considérons la règle d'inférence $\frac{p}{Op}$ (dite Règle de Nécessité ou *RN*), qui signifie que si le système contient *p* (l'hypothèse), il contient $\mathbf{O}p$ (la conclusion). À partir de là, on peut dériver les formules suivantes. Les preuves sont données dans [Abou El Kalam & Balbiani 2009] :

- $\mathbf{O}\phi \wedge \mathbf{O}\psi \rightarrow \mathbf{O}(\phi \wedge \psi)$ (26)
- $\mathbf{O}\phi \wedge \mathbf{R}\psi \rightarrow \mathbf{R}(\phi \wedge \psi)$, (27)
- $\mathbf{O}\phi \wedge \mathbf{P}\psi \rightarrow \mathbf{P}(\phi \wedge \psi)$. (28)

À l'inverse, nous avons démontré dans [Abou El Kalam & Balbiani 2009] que les règles suivantes ne sont pas dérivables :

- $\mathbf{P}\phi \wedge \mathbf{P}\psi \rightarrow \mathbf{P}(\phi \wedge \psi)$**.** (29)
- $\mathbf{R}\phi \wedge \mathbf{P}\psi \rightarrow \mathbf{P}(\phi \wedge \psi)$, (30)
- $\mathbf{R}\phi \wedge \mathbf{R}\psi \rightarrow \mathbf{R}(\phi \wedge \psi)$**,** (31)

Prenons des exemples simples : même si les deux formules "il est permis d'ouvrir la porte : **P**(*ouvrir la porte*)" et "il est permis de fermer la porte : **P**(*fermer la porte*)", la formule "*il est permis d'ouvrir et de fermer la porte en même temps*" : **P**(*ouvrir la porte* $\wedge$ *fermer la porte*) n'a pas de sens. La formule (29) n'est donc pas dérivable. Des exemples similaires peuvent être donnés pour la formule (30).

Pour la formule (31), même si elle n'est pas dérivable dans notre langage *LR*, notre intuition nous pousse à l'accepter. Elle correspond au fait que si les formules ϕ et ψ sont recommandées séparément, alors elles sont recommandées conjointement.

Dans cette section, nous avons défini notre langage de recommandation RSL (pour *Recommendation*

[8] Nous avons utilisé ce symbole pour exprimer l'ensemble vide.

Specification Language) à travers sa syntaxe, sémantique et axiomatique. Celui-ci constitue à notre connaissance, le premier travail publié sur les recommandations. Bien évidement, il reste dans un état embryonnaire et nécessite des améliorations. D'ailleurs, dans [Abou El Kalam & Balbiani 2009], nous présentons des extensions qui, sous certaines conditions, enrichissent au fur et à mesure notre langage.

VI. Conclusions

A l'image de la plupart des applications utilisant les nouvelles technologies de communications, les infrastructures critiques cherchent à s'ouvrir, utiliser des réseaux à base d'IP, partager, collaborer, etc. Néanmoins, ceci peut causer de sérieux problèmes de sécurité. La question évidente qui se pose est : comment gérer de manière sécurisée, dans un environnement de suspicion et d'hostilité, des applications complexes, distribuées, multi-organisationnelles, mais qui souhaitent intéropérer et partager des ressources tout en gardant son autonomie et sa flexibilité ? La réponse à ce dilemme devrait passer par une démarche rigoureuse de sécurité incluant notamment l'expression et la formalisation des politiques de sécurité. Traitant de ce sujet, ce premier chapitre contient deux grandes parties.

La première propose le cadriciel (*framework*) PolyOrBAC qui se base globalement sur les principes suivants :

- l'expression des politiques locales de chaque organisation par des règles OrBAC
- chacune des organisations souhaitant rendre accessibles ses ressources crée des services Web (SW) dédiés et les publient. Grâce aux technologies des SW, non seulement les communications dans un environnement hétérogène seront facilitées, mais aussi les détails internes à chaque organisation (structures, implémentations, plate formes, etc.) resteront privés.
- L'utilisation d'un ou plusieurs services est formalisée par un contrat électronique qui sera traduit par un automate au niveau de chaque organisation. Cet automate exprime les différents échanges autorisés, interdits et obligatoires. De plus, des règles de sécurité sont ajoutées au niveau de chaque organisation pour contrôler l'accès et l'utilisation des services. Afin d'exprimer ces règles dans les politiques locales, il faut trouver un moyen pour représenter les objets (utilisateurs d'un côté et service Web de l'autre) distants. Pour cela, nous avons proposé la notion d'image de SW qui représentera le vrai SW dans les règles locales de l'organisation cliente, et la notion d'utilisateur virtuel qui représentera en quelque sorte les utilisateurs distants dans la politique locale de l'organisation fournissant le service.
- En plus de la mise en œuvre des politiques OrBAC locales au niveau de chaque organisation, des techniques de vérification de modèles (*model-checking*) sur les automates représentant les contrats sont vérifiés en temps réel. Le but étant de détecter les abus ou les violations éventuelles des contrats établis.

La deuxième partie de ce chapitre modélise le concept de recommandations. En effet, malgré sa grande richesse, PolyOrBAC, comme l'ensemble des modèles de sécurité, ne permet pas de formaliser et de raisonner sur les recommandations, alors que celles-ci deviennent omniprésentes dans les règlements de sécurité. Pour pallier ce manque, nous sommes tout d'abord partis du fait qu'une recommandation peut être vue comme une modalité plus forte que les permissions et moins forte que les obligations. Ensuite, nous avons proposé un nouveau formalisme pour la spécification des

recommandations. Moyennant quelques ajustements, ce langage peut être intégré à PolyOrBAC et à tout autre langage basé ou inspiré de la logique déontique.

Enfin, notons qu'une grande partie de l'activité présentée dans ce chapitre a été soutenue par le projet CRUTIAL et a été partiellement traitée dans le cadre de la thèse d'Amine Baïna que j'ai co-encadré. Les différentes facettes de ce travail (PolyOrBAC, complexité d'OrBAC, recommandations) ont donné lieu à plusieurs publications internationales notamment :

- un article de revue spécialisée dans la protection des infrastructures critiques [Abou El Kalam *et al.*, 2009a],
- un chapitre de livre sur la sécurité des services Web [Abou El Kalam & Deswarte, 2009b],
- plusieurs publications dans des conférences internationales avec actes et comité de lecteurs [Abou El Kalam 2008b, Abou El Kalam *et al.*, 2007a, Abou El Kalam *et al.*, 2007b, Abou El Kalam & Deswarte, 2006a, Baïna *et al.*, 2008a, Baina *et al.* 2008b, Abou El Kalam 2008a, Abou El Kalam & Balbiani 2009],
- deux rapports de contrat [Abou El Kalam *et al.*, 2007b, A. Abou El Kalam et al. 2008e]

Ce travail a ouvert de nouvelles thématiques de recherche que nous sommes en train de développer davantage avec d'autres collègues, notamment à travers un projet avec Airbus sur la dérivation de politiques de sécurité à partir de mécanismes de surveillances réseau. Un financement de la DPAC m'est accordé sur ce sujet pour un post-doc et une thèse dont la date prévue de démarrage est fin décembre 2009.

Chapitre 2 : Analyse de politiques de sécurité

I. Motivation

Le principal atout de l'utilisation d'une approche formelle dans la spécification d'une politique de sécurité réside dans l'élimination d'un certain nombre des ambiguïtés de la spécification, et d'aider l'administrateur à spécifier, définir et formaliser la politique de sécurité. Les profits en sont multiples : l'analyse des problèmes d'interopérabilité entre plusieurs politiques, l'interrogation de la politique par des requêtes concernant les accès, ou la manipulation de la spécification par des transformations mathématiques et avec l'assistance d'outils de preuve, notamment pour vérifier la cohérence de la politique de sécurité. Ceci est particulièrement important dans les infrastructures critiques qui, comme nous l'avons expliqué dans le chapitre précédent, sont multi-organisationnelles et dynamiques (les droits des utilisateurs doivent pouvoir varier selon le *contexte* courant). Dans les sous-sections suivantes, nous détaillons quelques types de raisonnements qu'on peut faire sur une politique de sécurité.

A. Consultation d'une politique de sécurité

Une politique de sécurité peut faire l'objet de plusieurs requêtes du type :

- Étant données certaines caractéristiques contextuelles, quelles sont pour un utilisateur jouant un certain rôle, les permissions (interdictions, obligations ou recommandations) de réaliser une certaine action sur un objet donné ?
- Qui a des privilèges (et lesquels) sur un objet donné ?
- Dans quel contexte tel utilisateur a-t-il tel privilège sur tel objet ?
- Qui peut déléguer quel privilège à un individu jouant un rôle donné, dans un contexte donné ?

B. Propriétés attendues d'une politique de sécurité

Il est nécessaire de s'assurer qu'avec une politique de sécurité définie et cohérente, il n'existe pas de situation dans laquelle un utilisateur peut violer une propriété de sécurité exigée. Par exemple : il peut apprendre, créer, modifier ou détruire une information alors qu'il n'a pas l'autorisation de la connaître. Ainsi, il peut être intéressant d'identifier les éléments de la politique provoquant le non respect des propriétés de sécurité souhaitées, c'est-à-dire les sous-ensembles minimaux de règles qui génèrent l'incohérence dans certaines situations.

Par ailleurs, la modélisation formelle d'une politique de sécurité demeure primordiale si l'on souhaite vérifier que l'implémentation du système d'information, et en particulier des mécanismes de contrôle d'accès, permet bien de garantir les propriétés de sécurité souhaitées.

C. Complétude et interopérabilité

L'objectif d'une politique de sécurité est de définir les règles à respecter pour protéger le système contre les menaces identifiées lors de l'analyse des risques. Le problème de la complétude d'une politique de sécurité peut être vu comme celui de l'exhaustivité du règlement correspondant. Ceci peut être vérifié en montrant qu'il existe une règle spécifiée dans la politique de sécurité qui définit la conduite à tenir face à chaque risque identifié.

D. Cohérence d'une politique de sécurité

Globalement, on peut distinguer quatre types d'incohérences dans la politique de sécurité :

- Il peut s'avérer que certains objectifs de sécurité soient contradictoires les uns avec les autres.
- Il est parfois possible que les règles de fonctionnement du système entrent en conflit avec les objectifs et les règles de sécurité qui ont été définis.
- Étant donné que les règles de sécurité permettent de savoir comment un état de sécurité peut évoluer, et que les objectifs de sécurité permettent de savoir si un état est sûr, il est a priori souhaitable que l'on puisse vérifier s'il n'est pas possible, partant d'un état sûr et en appliquant les règles de sécurité, d'atteindre un état non-sûr (c'est-à-dire, un état qui compromet l'un des objectifs de sécurité). Notons que ce type de vérification peut être difficile, car il correspond à la résolution du problème de protection, indécidable dans le cas général [Harrison *et al*. 1976].
- Enfin, deux ou plusieurs règles de sécurité présentes dans la politique peuvent elles-mêmes se contredire ; c'est typiquement le cas où l'on a, par exemple, une règle qui recommande à un certain utilisateur la réalisation d'une certaine action, tandis qu'une autre règle le lui interdit. C'est ce dernier type d'incohérence qui nous intéresse dans ce chapitre.

Notons que l'ensemble des problématiques (complétude, cohérence, etc.) que nous avons identifié dans cette section s'imposent fortement si l'on souhaite faire interopérer deux organisations dotées chacune de sa propre politique de sécurité. Des problèmes plus ou moins analogues se posent lors de la fusion de plusieurs politiques de sécurité, par exemple, dans le cadre d'une restructuration entre deux organismes. Dans ces cas, il serait intéressant de détecter les conflits dans la politique obtenue globale, puis la proposition d'une méthode permettant de résoudre ces conflits.

Dans ce chapitre, nous proposons l'utilisation de la programmation logique par contraintes (PLC), pour pouvoir effectuer des raisonnements et vérifications sur les modèles de sécurité, notamment le modèle OrBAC (et donc PolyOrBAC). En particulier, la PLC peut être utilisée, d'abord pour spécifier les règles de fonctionnement et de sécurité du système, puis pour interroger la politique et détecter et résoudre les conflits éventuels.

II. La programmation logique par contraintes (PLC)

La programmation logique (*PL*) est un paradigme de programmation qui a été introduit par R. Kowlski [Kowlski 1974] et A. Clomerauer [Clomerauer 1990]. Elle est basée sur un sous-ensemble de la logique des prédicats du premier ordre (les clauses de *Horn*). Plusieurs raisons sont à l'origine de notre choix de la *PL* :

- Sa déclarativité qui permet de décrire les connaissances d'un problème d'une façon très simple.

- Sa puissance grâce notamment aux processus d'unification et de résolution pour inférer la solution du problème à résoudre, à partir de sa description.
- Sa souplesse car en comparant avec les langages procéduraux par exemple, dans la *PL,* une procédure peut être utilisée pour résoudre différents types de problèmes suivant l'instantiation de ses arguments.

En 1986, la *PL* a été étendue par J. Jaffar et J. L. Lassez. Cette extension a donné naissance à la *programmation Logique par contraintes* (*PLC*) [Jaffar & Lassez 1987]. La *PLC* constitue une combinaison entre le processus de déduction logique et un ensemble d'algorithmes incrémentaux de résolution de contraintes (dits aussi, *solveurs* de contraintes) [Hentrayck 1989].

L'utilisation de la PLC donc notre contexte peut tout naturellement servir à spécifier les règles de fonctionnement du système ; ceci se fait de manière similaire à une représentation en logique classique. Par exemple, dans OrBAC, on peut tout à fait utiliser les prédicats : *Habilite*(), *Appartient*(), *Réalise*(), *etc*. Ainsi; le fait que l'hôpital *Purpan* habilite le sujet *Bob* dans le rôle de directeur est représentable dans la *PLC* par *Habilite*(*Purpan*, *Bob*, *Directeur*).

À ce stade, il est également important de définir les fonctions dont nous aurons besoin par la suite ; par exemple, on peut considérer la fonction *Patient*(*Purpan*) qui retourne la liste des patients de *Purpan*. La spécification des différentes contraintes fonctionnelles, notamment l'exclusion mutuelle entre les rôles, se fait de manière classique, par exemple, la règle $\forall s \in Sujets \ \neg[Habilite(Purpan, Chirurgien, s) \wedge (Habilite(Purpan, Anesthésiste, s)]$ signifie qu'au sein de l'hôpital *Purpan*, aucun sujet ne peut jouer à la fois les rôles chirurgien et anesthésiste.

Dans le même sens, l'interrogation de la politique de sécurité dans la *PLC* se fait de manière classique. Par exemple, si l'administrateur souhaite savoir qui est recommandé à lire une notice, il n'a qu'à interroger le moteur Prolog avec une requête du type $\exists n, Notice(n) \wedge \mathbf{R}(x, Read, n)$. La réponse à de telles requêtes prend généralement l'une des deux formes suivantes : le système liste les personnes ayant la recommandation de lire la notice ; ou alors renvoie une formule qui correspond aux conditions qui satisfont la requête. Cette seconde technique est connue sous le terme réponse intentionnelle [Cholvy & Demolombe 1986].

III. Utilisation de la PLC pour la résolution de conflits

Dans cette section, on s'intéresse au problème de résolution de conflits dans les politiques de sécurité. Ce type de problème a fait l'objet de plusieurs travaux antérieurs, notamment pour RBAC [Strembeck 2004], OrBAC [Benferhat *et al.* 2003] et les bases de données [Bertino *et al.* 1996]. L'approche que nous suggérons s'appuie sur la programmation logique par contraintes (PLC). Mais avant de rentrer dans les détails, notons tout d'abord que plusieurs définitions ont été données à la notion de conflit. La RFC 3198 précise qu'un conflit de politiques survient quand les actions de deux règles se contredisent [RFC 3198]. Moffett et Lupu [Moffett & Sloman 1993, Lupu & Sloman 1999] distinguent deux types de conflits : conflits de modalités (contradiction entre des types de règles), et conflits applicatifs (dû à une incohérence au niveau du fonctionnement ou à des limites du modèle).

Dans ce travail, nous nous intéressons aux conflits de modalités, et plus précisément aux situations dans lesquelles un utilisateur aurait simultanément la permission (ou l'obligation ou la recommandation) et l'interdiction d'effectuer une action sur un objet. De manière formelle, ceci revient

à chercher les situations où au moins une des formules suivantes est vraie : $\mathbf{R}p \wedge \mathbf{F}q$, $\mathbf{P}p \wedge \mathbf{F}q$, $\mathbf{O}p \wedge \mathbf{F}q$; et par extension, on peut également considérer les situations où les formules suivantes sont vraies : $\mathbf{I}p \wedge \mathbf{R}q$ et $\mathbf{I}p \wedge \mathbf{O}q$. Cette situation n'est pas tout à fait impossible dans la mesure où, à un moment donné, on peut avoir plusieurs règles (permissions, interdictions, recommandations, obligations) pour un même 5-uplet donné (*Organisation, Rôle, Activité, Vue, Contexte*). En effet, outre le fait qu'un conflit peut tout simplement avoir lieu entre les règles existantes (règles figurant dans la base des faits), il peut également survenir suite à certains événements, par exemple :

– lors d'une intervention temporaire de l'administrateur pour ajouter ou modifier une règle ou n'importe quelle autre entité de la base des faits ;

– lors de la dérivation de règles suite à l'application des mécanismes d'héritage (propagation « par héritage ») et des mécanismes logiques de déduction (en exploitant l'axiomatique du langage, par exemple en remplaçant toute recommandation par une permission).

Afin de remédier à ce type de problèmes, une vérification – hors ligne – de la cohérence de la politique de sécurité s'impose ; autrement dit, vérifier que le système est dans un état sûr. Pour cela, on peut utiliser des techniques comme la méthode des tableaux [Fitting 1993]. Il faut également s'assurer que toute intervention temporaire de l'administrateur pour ajouter ou modifier une règle ou une entité laisse la politique dans un état de cohérence.

Pour arriver à ces fins, nous associons des priorités (un entier naturel) aux règles. Ainsi une règle classique OrBAC par exemple (Tableau 10a) sera remplacée par celle du Tableau 10b.

Tableau 10a : Règle OrBAC classique

Permission (*org*, *r*, *v*, *a*, *c*) ∧
Habilite (*org*, *s*, *r*) ∧
Considère (*org*, *α*, *a*) ∧
Utilise (*org*, *o*, *v*) ∧
Définit (*org*, *s*, *α*, *o*, *c*)
→ *Est_ermis*(*s*, *α*, *o*)

Tableau 10b : Règle OrBAC avec priorités.

Permission (*org*, *r*, *v*, *a*, *c*, <u>*Priorité*</u>) ∧
Habilite (*org*, *s*, *r*) ∧
Considère (*org*, *α*, *a*) ∧
Utilise (*org*, *o*, *v*) ∧
Définit (*org*, *s*, *α*, *o*, *c*)
→ *Est_permis*(*s*, *α*, *o*, <u>*Priorité*</u>)

Nous avons ensuite proposé et implémenté avec Prolog l'algorithme suivant pour détecter et résoudre les conflits. L'idée de base est de partir d'un état sûr du système et de surveiller ensuite toute intervention de l'administrateur (e.g., pour ajouter une règle). Ce type d'actions déclenche automatiquement le processus suivant :

1. la nouvelle règle (notons la *Permission* (*org*, *r*, *v*, *a*, *c*)) est maintenue dans une base temporaire ;
2. le serveur d'autorisation invoque le processus qui recherche et extrait les règles en relation avec cette nouvelle règle. Il s'agit des règles ayant le même 5-uplet (*org*, *r*, *v*, *a*, *c*) ; par exemple *Interdiction* (*org*, *r*, *v*, *a*, *c*) ;
3. si un conflit est détecté[9], le système identifie et liste les éléments de la politique (règles) à l'origine du conflit (soit les sous-ensembles minimaux de règles générant ce conflit) ;

[9] Rappelons que nous considérant comme conflictuelle, toute situation où l'une des formules suivantes est vraie : $\mathbf{R}p \wedge \mathbf{F}q$, $\mathbf{P}p \wedge \mathbf{F}q$, $\mathbf{O}p \wedge \mathbf{F}q$, $\mathbf{I}p \wedge \mathbf{F}q$, $\mathbf{I}p \wedge \mathbf{R}q$ ou $\mathbf{I}p \wedge \mathbf{O}q$

4. Pour chacune de ces règles, on déduit les décisions (selon Tableau 10b) du type *Est_permis*(*s*, *α*, *o, Priorité*), *Est_interdit*(*s*, *α*, *o, Priorité*), etc.

5. Pour chaque (*s*, *o*, *a*), on considère la décision de permission (à partir des *Est_permis*(*s*, *α*, *o, Priorité*) déduits de l'étape précédente) qui a la plus grande priorité ; puis on fait la même chose pour les *Est_interdit*(*s*, *α*, *o, Priorité*), *Est_recommandé*(*s*, *α*, *o, Priorité*) et *Est_obligatoire*(*s*, *α*, *o, Priorité*) ;

6. Entre ces quatre prédicats (qui correspondent à des décisions d'accès avec des priorités), on retient celui qui a la plus grande priorité. En cas d'égalité, on considère que *Est_interdit* est plus critique que *Est_obligatoire*, qui à son tour est plus critique que *Est_recommandé*, qui enfin, est plus critique que *Est_permis.*

7. le système propose des corrections en invitant l'utilisateur à reformuler l'une des règles impliquées, à modifier un objectif de sécurité ou une règle de fonctionnement, ou à changer la priorité de la règle à ajouter.

8. Le système prend ces corrections en compte et revérifie la cohérence de la politique de sécurité. Si le conflit est résolu, la nouvelle règle est sauvegardée dans la base des faits des règles, sinon, on reprend depuis l'étape 2.

Notons que ce travail a été implémenté en utilisant le langage Prolog (*Programmation Logique*) [Colmerauer 1990], le langage le plus répandu de la *PLC*. Ce langage a été conçu et développé à Marseille dans le courant des années soixante-dix par P. Roussel et A. Colmerauer. Il permet d'énoncer les connaissances d'un problème d'une manière déclarative tout en fournissant un mécanisme de résolution. Prolog utilise la syntaxe de la logique et sa sémantique pour décrire les connaissances d'un domaine. Un programme logique est défini à l'aide d'un ensemble d'objets et de relations qui formalisent le problème à résoudre. Il est constitué d'un ensemble de clauses ; une clause est une affirmation portant sur des atomes logiques exprimant une relation entre des termes ; les termes sont les objets de l'univers.

IV. Conclusions

La spécification des politiques de sécurité à travers des modèles ne s'avère vraiment intéressante que si l'on dispose de moyens pour raisonner sur cette politique et pouvoir l'interroger. Plusieurs techniques et cadres logiques ont été proposés dans la littérature pour assurer cette tâche. Nos travaux dans ce domaine consistaient à proposer et implémenter une solution simple pour résoudre les conflits de modalités dans les politiques de sécurité. Basée sur la Programmation Logique par Contraintes (PLC), notre solution associe des priorités aux règles. Concrètement, un conflit est détecté quand on arrive par les processus de déduction logique à des situations où on a, pour le même tuplet (*s*, *o*, *a*), des décisions différentes (par exemple, à la fois *Est_recommandé*(*s*, *o*, *a, priorité*1) et *Est_interdit*(*s*, *o*, *a, priorité*2). La priorité est utilisée pour statuer de la règle à appliquer et de la décision d'accès à prendre.

Chapitre 3 : Mise en œuvre de politiques de sécurité et de QoS

I . Motivation

Dans le premier chapitre, nous avons exprimé les besoins de sécurité des systèmes qui nous intéressent, à savoir les infrastructures critiques, et nous avons proposé PolyOrBAC, un cadriciel (*framework*) qui couvre notamment la richesse de ce type de système. Celui-ci est générique et peut être appliqué de manière globale aux applications interopérables, hétérogènes, distribuées, dynamiques et multi-organisationnelles. Néanmoins, PolyOrBAC ne décrit pas comment gérer les recommandations, alors que cette modalité d'accès semble présente dans la plupart des règlements de sécurité. Nous avons donc proposé un langage qui spécifie de manière formelle la syntaxe, la sémantique, l'axiomatique ainsi que les règles d'inférences liées aux recommandations.

Une fois spécifiée, une politique de sécurité peut être utilisée notamment pour des raisonnements logiques. Dans le deuxième chapitre, nous avons proposé notre approche pour gérer les conflits de modalités dans les politiques de sécurité qui contiennent des permissions, recommandations, obligations, mais aussi des interdictions et des formules déconseillées.

Dans la pratique, après avoir spécifié une politique de sécurité et vérifié sa cohérence, il faut bien évidement la mettre en œuvre par des mécanismes appropriés de sécurité, par exemple au niveau des règles de pare-feux ou des politiques de réseaux privés virtuels, connus sous le nom VPN (pour *Virtual Private Network*). Le déploiement de politique de contrôle d'accès, notamment OrBAC, ont été décrits dans plusieurs travaux, notamment [Cuppens-Boulahia 2007]. Dans ce rapport, nous nous intéressons aux VPN.

Avant de détailler notre solution, commençons par la situer dans son contexte. Toujours dans le cadre d'infrastructures et réseaux critiques, nous nous sommes intéressés aux nouvelles architectures des réseaux avioniques embarqués que nous avons étudiées dans le cadre du projet ADCN (pour *Advanced Data Communication Networks*) [Abou El Kalam 2008c, Abou El Kalam 2008d, Abou El Kalam 2009] . En effet, les avions actuels et futurs ont une architecture totalement nouvelle qui intègre différents domaines, applications et réseaux hétérogènes. Le problème de sécurité s'impose ainsi de manière forte dans ce type d'applications critiques. Dans cette optique, nous avons commencé par en étudier les besoins de sécurité, puis proposer un nouveau type de passerelles embarquées où une grande partie de la politique de sécurité sera déployée. Plus précisément, dans les nouvelles architectures avioniques, nous pouvons distinguer trois domaines :

- le domaine *avionique* (dit *Dark Green)* : contient les activités vitales durant un vol comme les fonctions de pilotage et de navigation, les commandes des ailes et du train d'atterrissage, les fonctions de calcul de poussée et de monitoring des équipements critiques, la communication radio par satellite ou VHF, etc.
- le domaine *Ethernet avionique* (dit *Light green)* : contient les fonctions de maintenance ainsi

que les informations des compagnies aériennes. Dans ce domaine, les sociétés de maintenance et les compagnies aériennes peuvent connecter directement leurs systèmes d'information (e.g., à travers leurs ordinateurs portables) à l'avion pour récupérer, traiter, télécharger ou recevoir certaines données à travers des applications dédiées.

– le domaine des *passagers* : contient les accès Internet (Internet embarqué), le WiFI, la vidéo à la demande (VoD), la radio, les jeux en réseau, les informations sur le vol (dites aussi MMS pour *Moving Map Systems*), etc.

Même si ces trois mondes intègrent des applications (COTS[10], applications de maintenance, etc.) et réseaux (AFDX, CAN, Ethernet, IP) différents, ils doivent coopérer et échanger des informations, souvent de sensibilités et criticités variables.

Malheureusement, les architectures des passerelles classiques ne s'adaptent pas à ces cas de figures, notamment pour tenir compte des différentes contraintes liées aux réseaux et applications. En effet, même en échangeant des informations sur le même réseau embarqué, il ne faut pas que les applications les moins critiques gênent le fonctionnement des plus critiques, que ce soit en terme de criticité liée à la sécurité (ex. protection contre les attaques compromettant la confidentialité, l'intégrité et la disponibilité) ou de qualité de service (ex. contraintes temporelles). Cette problématique nécessite donc l'étude, l'analyse, la simulation et l'évaluation des différents flux (traversant des passerelles) entre plusieurs réseaux de criticités différentes, pour pouvoir proposer une plate-forme sécurité et qualité de service (QoS) adaptative aux contraintes de l'environnement d'exécution et des réseaux cibles.

Pour répondre à ces besoins et relier les différents domaines de l'avion (notamment les domaines *light green* et *dark green)*, nous avons proposé et implémenté la passerelle de sécurité de la figure 12. Celle-ci intègre, à la fois, des mécanismes de sécurité et de QoS comme nous montrons dans la suite de cette section [Abou El Kalam 2009, Mostafa *et al.* 2009].

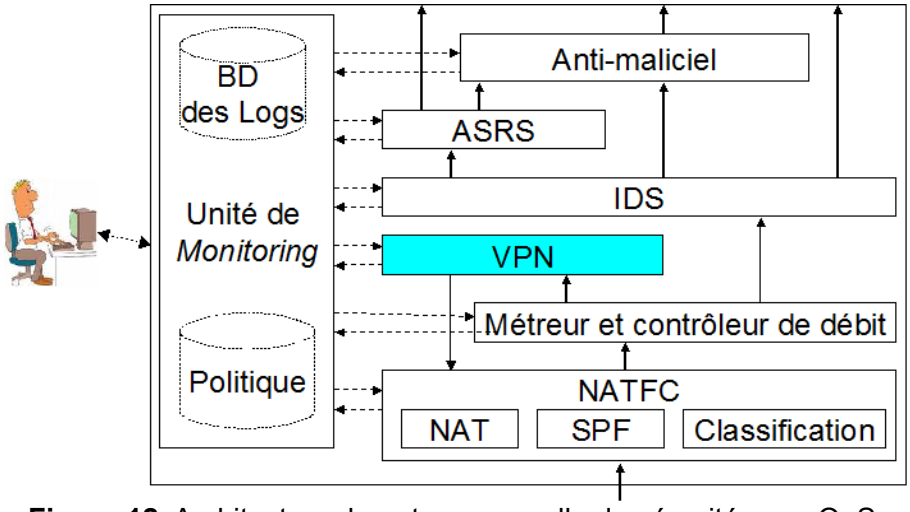


Figure 12. Architecture de notre passerelle de sécurité avec QoS.

Même si dans ce rapport nous avons fait le choix de ne détailler que l'un des modules de notre passerelle, à savoir le module VPN, nous présentons tout d'abord très succinctement et de manière globale les principaux composants :

[10] *commercial off-the-shelf* ou *COTS* est utilisé pour désigner un composant sur étagère, souvent fabriqué en grande série

– Le *NATFC*, pour *Network Address translation, Filtering, and Classification* : reçoit les paquets entrants et offre trois fonctions principales : traduction d'adresses (dite *NAT*, pour *Network Address Translation*), pare-feu de filtrage et classification QoS. Traditionnellement, ces fonctions sont réalisées séparément. Dans notre cas, étant donné que ces trois fonctions dépendent fortement des algorithmes de classification pour identifier des flux à laquelle appartient le paquet, nous les avons fusionnées en une seule unité pour améliorer les performances. À cet égard, nous ne réitérons pas l'appel de la fonction de classification pour chaque paquet, celle-ci est invoquée une seule fois, et le résultat est utilisé pour les trois fonctions : traduction NAT, filtrage pare-feu et classification QoS. Nous avons ainsi réimplémenté les processus de ces trois fonctions et nous avons modifié la structure des tables du pare-feu *PF* (*Packet Filter*) de NetBSD. Nos expérimentations ont abouti à des résultats encourageants et à une diminution non-négligeable du temps de traitement [Mostafa *et al.* 2009].

– *Métreur et contrôleur de débit* (*Meter and rate controller unit*) : afin de surveiller et contrôler la stabilité du trafic et garantir un partage équitable de la bande passante du réseau, nous suggérons d'utiliser un métreur (*meter*) ainsi qu'un contrôleur de débit (*rate controller*). Plus précisément, le métreur est utilisé pour calculer certains paramètres comme la bande passante, tandis que le contrôleur de débit statue sur les flux à accepter ou rejeter selon la politique de QoS prédéfinie. Ces composants sont placés juste après le NATFC, étant donné qu'on ne souhaite assurer le suivi et le contrôle que des flux acceptés après la classification.

– *Intrusion Detection System* (*IDS*) : il s'agit de systèmes de détection d'intrusions adaptés au contexte avionique, notamment en terme d'algorithmes de détection et de protocoles de décodage et éventuellement de décompression des données de charge utile.

– *Application Specific Rules and Signatures* (*ASRS*) : alors que l'APF (*Application Proxy Firewall*) peut fournir un bon niveau de sécurité, il ajoute un surcoût (*overhead*) notable dans le contexte avionique, notamment en raison de la destruction des entêtes et leur reconstruction. Pour remédier à ce type de problèmes, nous suggérons d'utiliser un *ASRS*, ensemble de signatures et règles spécifiques qui doivent être appliquées et vérifiées pour garantir un traitement sécurisé des applications qui nous concernent.

– *Anti-maliciel* (*Anti-malware scanner*) : est utilisé pour inspecter les fichiers téléchargés pour la détection des *maliciels* tels que des virus, vers, ou chevaux de Troie.

– *Unité de monitoring* (*Monitoring and adaptive unit*) : surveille l'état interne de la passerelle et applique les mesures d'adaptation si nécessaire. Le suivi tient compte de la politique de sécurité et des alarmes et autres informations des fichiers journaux. Outre l'analyse de certaines activités, cette unité peut également recevoir des informations sur l'état du réseau protégé et gérer la coopération entre les différentes unités de la passerelle.

– *Interface administrateur*: interface graphique permettant aux administrateurs de définir des politiques et exécuter des tâches d'audit. Elle doit être extensible et suffisamment souple pour permettre l'ajout de règles ou de nouvelles signatures.

– *Module du réseau privé virtuel (Virtual Private Network*) : si le paquet est chiffré ou authentifié, il est envoyé au module VPN pour les traitements cryptographiques.

Dans ce rapport, nous ne détaillons que le module VPN. Bien évidement, dans notre contexte, celui-ci doit tenir compte, non seulement des exigences de sécurité, mais aussi de QoS. En effet, proposer

une solution qui assure la sécurité au détriment de la QoS, ou l'inverse, serait tout simplement rejetée par les professionnels de ce domaine. Par exemple, des mécanismes de sécurité qui retardent l'envoi ou la réception des commandes de vol dans un avion, ou de manière moindre, les commandes d'un système électrique dans une grille d'électricité, seraient inacceptables. Malheureusement, la sécurité n'est pas gratuite et demande souvent plus de ressources et augmente les délais de transmissions et les temps de traitements. Les applications temps réel ont besoin d'un traitement spécial pour parvenir à leurs objectifs et en général, souffrent considérablement de délais non-acceptables. La Qualité de Service intervient ainsi pour remédier à ce genre de problèmes et gérer plus efficacement le trafic en fonction de ses besoins spécifiques.

Plus grave encore, dans les applications qui nous intéressent (ayant des contraintes temporelles fortes), si les mécanismes de sécurité cachent (chiffrent ou rendent indisponibles) les informations nécessaires pour mettre en œuvre la QoS, ils seront tout simplement rejetés. C'est typiquement le cas d'IPSec (*IP Security Protocol*) qui chiffre des données nécessaires à la QoS.

L'idée est donc de développer des mécanismes qui répondent à la fois aux besoins de sécurité et de QoS. Pour arriver à ces fins, nous avons fait le choix d'améliorer IPSec pour prendre en compte la QoS. Le choix d'IPSec n'est pas dû au hasard, mais mûrement réfléchi. En effet, IPSec est la norme de sécurité au niveau d'IP, le réseau le plus déployé au monde. D'ailleurs, IP est actuellement présent dans les nouvelles architectures avioniques qui, il y a quelques années, se limitaient à des réseaux comme l'ARINC-664 (e.g., AFDX), CAN, ARINC 429 et FlexRay. De plus, IPSec est proposé dans quasiment toutes les offres commerciales de VPN (*Virtual Private Network*) pour assurer la protection des échanges IP, et ce, quel que soit le type de réseau de transport exploité dans l'interconnexion (MPLS, IP, ...). Par ailleurs, étant donné qu'IPSec (*Internet Protocol Security*) est conçu pour fournir des services de sécurité (contrôle d'accès, authentification de l'émetteur, anti-rejeu, intégrité non connectée et confidentialité des données) au niveau de la couche IP (Ipv4 et IPv6), en dessous de la couche de transport, il reste transparent pour les applications et les utilisateurs ; il n'est donc pas nécessaire de changer les couches applicatives lorsque le protocole IPSec est implémenté au niveau des passerelles de sécurité.

Cependant, pour des raisons de sécurité, le protocole ESP (*Encapsulating Security Payload*) d'IPSec cache les entêtes TCP/IP dans la charge utile chiffrée, empêchant ainsi des équipements de contrôle réseau tels que les routeurs et les commutateurs de réaliser la classification QoS. De manière globale, Quel que soit la méthode de QoS utilisée (*Integrated service* [Braden *et al.* 1994] ou *Differentiated service* [Blake *et al.* 1998]), la classification QoS (utilisant le concept de Classe de Service) sépare le trafic réseau en classes distinctes et fournit, pour chaque paquet, un service dépendant de la classe à laquelle il appartient. Plus concrètement, une valeur de priorité est assignée à chaque paquet. Cette priorité est stockée dans le champ *Type of Service* (ToS) de l'entête IPv4. Si on prend l'architecture à service différencié (*DiffServ*) comme exemple, cette valeur de priorité est appelée *Differentiated Service Code Point* (DSCP) [Nichols *et al.* 1998].

Dans les architectures QoS, le classificateur, dit *Multi-Field (MF) classifier* inspecte plusieurs champs de chaque paquet entrant et le classe [Borg *et al.* 1999]. Nous avons donc commencé par voir quels sont exactement les champs inspectés, c'est-à-dire ceux dont on a besoin pour faire la QoS. Le but étant de voir si aucun de ces champs n'est chiffré lors de l'application d'IPSec ; si c'est le cas, voir si on peut les rendre disponibles au classificateur sans pour autant compromettre la sécurité.

Dans la pratique, comme nous verrons plus loin, IPSec ESP en mode tunnel (le mode le plus utilisé pour les VPN) empêche la QoS. Afin de résoudre ce problème, nous avons proposé une amélioration d'IPSec que nous avons baptisé Q-ESP, pour *QoS-friendly Encapsulating Security Payload* (*Q-ESP*). Avant de décrire Q-ESP, nous présentons brièvement les éléments de base nécessaires à la compréhension de notre travail, à savoir le protocole IPSec et les mécanismes QoS [Abou El Kalam *et al.*, 2009b, Mostafa *et al.* 2008b].

II. Éléments de base

A. *IPSec*

IPSec est un protocole de la couche 3 du modèle OSI. Développé et normalisé par l'IETF, il est intégré de base à Ipv6 et porté vers Ipv4 [Ken & Atkinson 98a, Ken & Atkinson 98b, Ken & Atkinson 98c]. La version actuelle de l'IPSec est composée des éléments suivants :

– deux protocoles de sécurité : AH (*Authentication Header*) qui assure l'intégrité des datagrammes et l'authentification de la source ; et ESP (*Encapsulating Security Payload*), qui assure la confidentialité des données ;

– des associations de sécurité ou *Security Association* (SA), qui représentent l'ensemble des services et paramètres de sécurité négociés entre deux hôtes, deux passerelles de sécurité ou un hôte et une passerelle ;

– des algorithmes pour l'authentification et le chiffrement.

IPSec AH et IPSec ESP peuvent être appliqués seuls ou en association. Chaque protocole peut fonctionner dans l'un des deux modes : le mode *Transport* ou le mode *Tunnel*. Dans le mode Transport, les mécanismes de sécurité ne sont appliqués qu'aux données de la couche supérieure ; les informations relevant d'opérations de la couche IP, telles que les contenus des en-têtes IP, ne sont donc pas protégées. À l'inverse, dans le mode tunnel, les données de la couche supérieure de même que les en-têtes IP sont protégées grâce à une encapsulation du paquet protégé dans un autre paquet IP.

Plus précisément, AH assure *l'authentification de l'origine* des données ainsi que leur *intégrité* pour les datagrammes IP. Il est possible, avec AH, de sélectionner la *détection de rejeux* en tant que service optionnel. Les principaux champs de AH sont :

– *Index de paramètres de Sécurité* ou SPI (*Security Parameter Index*) : valeur aléatoire utilisée en association avec l'adresse IP destination pour identifier l'association de sécurité à laquelle appartient le datagramme.

– *Numéro de séquence* : valeur d'un compteur utilisée pour détecter les datagrammes IP rejoués afin d'assurer l'intégrité des séquences de datagrammes.

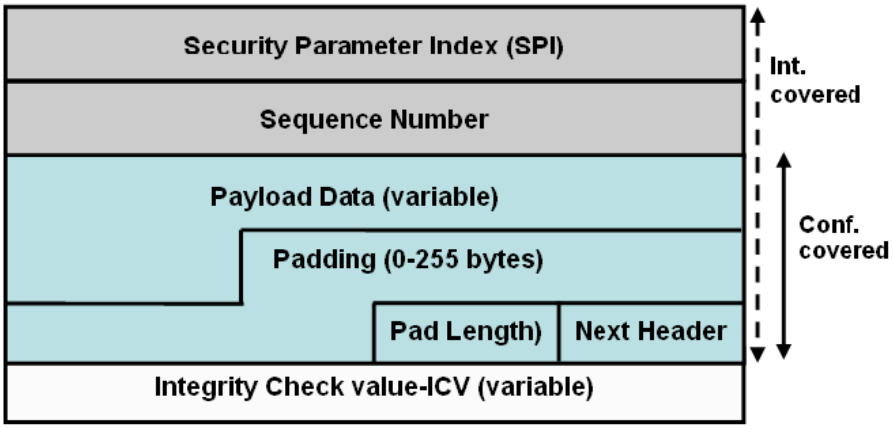
– *Données d'authentification* : valeur utilisée pour l'authentification et la vérification de l'intégrité du datagramme IP ; elle est calculée en appliquant une fonction de *hachage* sécurisée (e.g., H-MAC) aux champs[11] du datagramme IP.

Par ailleurs, comme indiqué dans la figure 13, l'en-tête ESP comprend les champs SPI et Numéro de séquence ainsi qu'un champ optionnel d'authentification des données. Le champ de charge utile

[11] Il ne s'agit que des champs non-modifialbles durant le transfert.

contient les données soumises à la protection de la confidentialité (chiffrement). Le bourrage permet d'obtenir, en ajoutant des bits aléatoires au champ charge utile, des données à chiffrer d'une taille en binaire qui est un multiple de la taille d'un bloc de chiffrement.

Figure 13 : Format de l'entête ESP.



Tout comme AH, le protocole ESP peut être appliqué en mode Transport ou en mode Tunnel. En mode Tunnel, tout le paquet IP est protégé. Pour cela, il est considéré comme un simple message, et un nouvel entête IP est créé (Figure 14), ce qui assure une certaine forme de confidentialité sur le trafic car il permet aux passerelles de sécurité de cacher l'identité des hôtes source et de destination ainsi que la taille réelle des datagrammes IP.

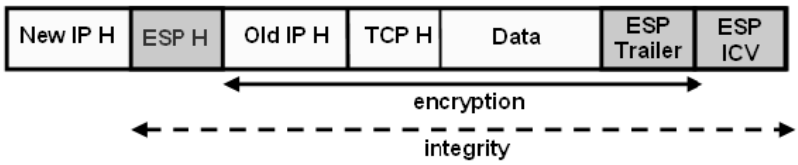


Figure 14. AH et ESP en mode tunnel.

B. *La QoS*

Différentes définitions ont été données à la QoS. Par exemple, l'UIT (*Union Internationale des Télécommunications*), la définit comme un "Effet global procuré par la qualité de fonctionnement d'un service qui détermine le degré de satisfaction de l'usager" ; de manière plus précise, l'IETF la définit comme étant la manière dont le service de livraison de paquets est fourni ; elle est généralement décrite par des paramètres tels que la bande passante, le débit de transfert d'informations (*Bulk Transfer Capacity*), la variation de délai (dite aussi la gigue ou le *Jitter* en anglais), les modèles de pertes de paquets (*Loss Patterns*) et éventuellement le ré-ordonnancement des paquets (*Packet Reordering*)".

Pour pouvoir mettre en place des garanties de QoS dans les réseaux IP, l'IETF a proposé deux modèles d'architecture basés sur des politiques de services différentes : *IntServ* et *DiffServ.*

Le modèle de réseau à intégration de services (IntServ) [Braden *et al.* 1994] est un modèle orienté flots (ou connexions), dans lesquels chaque flot peut faire une demande de QoS spécifique. La garantie

de la QoS de chaque flot est alors effectuée par un contrôle d'admission et une réservation préalable des ressources. Il s'agit d'une gestion préventive de la QoS, effectuée a priori lors de la réservation des ressources. À l'inverse, le modèle de réseau à différenciation de services (DiffServ) [Blake *et al.* 1998] agrège les flux en quelques grandes classes de trafics qui ont chacune leurs besoins spécifiques de QoS. La QoS est alors assurée au niveau des nœuds du réseau (les routeurs) par des traitements spécifiques à chaque classe de service. Ainsi, dans DiffServ, la différenciation de service n'est plus un modèle orienté "flot", mais un modèle orienté "nœud", chaque nœud du réseau pouvant avoir sa propre politique de traitement des paquets.

Par ailleurs, pour mettre en œuvre la QoS, quatre principaux modules ont été défini :

- *Classification du trafic* : permet d'identifier et de caractériser le trafic à l'intérieur des réseaux en associant chaque paquet à une classe. La classification se fait généralement selon certains champs des entêtes des couches 3 et 4.

- *Conditionnement du trafic* : après la classification d'un paquet, un nœud peut effectuer des opérations de contrôle de trafic pour s'assurer que le flux respecte le contrat. Nous distinguons quatre processus principaux du conditionneur de trafic : le métreur (*Meter,* en anglais) : mesure les caractéristiques temporelles des paquets sélectionnés par le classificateur et les compare avec le profil souscrit pour vérifier leur conformité ; le marqueur (*Marker*) : définit ou redéfinit la valeur du DSCP (*Differentiated Service Code Point*) en se basant sur les informations fournies par le métreur ; le lisseur (*Shaper*) : procède au lissage de trafic en le retardant de telle sorte qu'il ne dépasse pas le débit contractuel associé au profil défini dans le contrat avec l'usager ; et le suppresseur (*Dropper*) : élimine le trafic dépassant le débit contractuel associé au profil du contrat de service usager.

- *Évitement (ou prévention) de la congestion* : calcule l'évolution de la taille moyenne de la file d'attente et réagit en éliminant certains paquets ou en envoyant des informations à l'équipement en amont de la surcharge. En effet, lorsque la taille moyenne de la file d'attente dépasse un seuil prédéfini, la passerelle élimine ou marque les paquets avec une certaine probabilité (celle-ci dépend de la taille de la file d'attente).

- *gestion de la congestion* : après la classification et le contrôle de trafic, les paquets sont envoyés vers la ou les interfaces de transmission, puis mis dans les files d'attente jusqu'à ce qu'ils puissent être transmis. Le mécanisme qui décide l'ordre des paquets à transmettre est appelé ordonnancement ou discipline de service (*scheduler*). Celui-ci permet les traitements différenciés selon la classe de trafic.

III. Q-ESP

A. Format des paquets Q-ESP

Dans la section précédente, nous avons vu que la classification des paquets est à la base de tous les mécanismes de mise en œuvre de la QoS. En effet, pour gérer les ressources réseau et assurer la QoS, les "*Multi-Field packet classifiers*" classifient chaque paquet en prenant en considération plusieurs informations des entêtes IP/TCP, identifient ensuite le flux concerné par le paquet, et selon ces informations, fournissent enfin le service approprié dans les réseaux IP. Notons au passage que pour classifier un paquet, une priorité lui est assignée, celle-ci est enregistrée dans le champ ToS (*Type of Service*) de l'entête IPv4 (dit "*trafic Class*" dans IPv6).

En effectuant des tests, nous avons remarqué que lorsque nous avons des VPN basés sur des tunnels IPSec, les mécanismes de QoS ne fonctionnent pas. En inspectant le trafic, nous avons remarqué que, pour des raisons évidentes de sécurité, le protocole IPSec ESP chiffre les entêtes IP/TCP, empêchant ainsi les équipements réseaux (routeurs) de faire la classification et donc de fournir la QoS, puisqu'un élément fondamental, à savoir la classification, ne peut être effectué.

Pour pallier ce problème sans compromettre la sécurité, nous avons commencé par analyser de près le fonctionnement et l'implémentation des classificateurs (*Multi-Field packet classifiers*) afin d'identifier les informations nécessaires et suffisantes pour pouvoir différencier le trafic et effectuer la classification. Nous avons trouvé qu'il s'agit de cinq champs appartenant à des entêtes de différentes couches [Mostafa *et al.* 2008a, Mostafa *et al.* 2008b]:

- *Entête de niveau Transport* : le *MF packet classifier* inspecte les ports source et destination. Ces champs aident à identifier les applications s'exécutant sur TCP/UDP ; une liste des ports connus (les *well-known ports*) est maintenue par l'IANA [IANA 2009].

- *Entête de niveau Réseau* : trois champs sont inspectés dans cette couche, le champ adresse IP source, qui aide à identifier l'entité émettrice, le champ adresse IP destination, qui aide à identifier le système d'extrémité recevant les données, et enfin, l'identifiant de protocole qui est utilisé pour identifier le protocole de niveau transport utilisé.

Ainsi, pour pouvoir effectuer la classification, et donc la QoS, ce quintuplet (*five-tuple*) doit être accessible par le classificateur, ce qui n'est pas le cas avec IPSec ESP. Notre première idée était donc de copier les champs nécessaires au niveau de l'entête Q-ESP. Mais cela ne suffit pas, nous avons également estimé nécessaire d'ajouter un nouveau champ, nommé TLP (pour, *Transport Layer Protocol*) qui indiquera le protocole du niveau transport (TCP/UDP) encapsulé dans la charge utile. Notons que sans ce champ, le récepteur ne peut pas savoir si les paquets reçus sont protégés par Q-ESP (ou pas) et par conséquent, s'ils doivent être adressés au module Q-ESP (pour le traitement spécifique à Q-ESP) ou aux modules de traitement des couches TCP ou UDP.

Ainsi, l'entête Q-ESP contient les six champs suivants (Figure 15) : *source port*, *destination port*, *TLP* (*Transport Layer Protocol*), *Reserved* (champs réservés pour les futures recherches et qui doivent être mis à zéro), *Security Parameters Index*, et *Sequence Number.*

En plus de l'entête, les paquets Q-ESP contiennent la charge utile, le *trailer* ainsi qu'un champ pour la valeur de l'authentification (dit ICV, pour *Integrity Check Value*) (Figure 15).

La charge utile (*payload*) contient le protocole de transport de la couche supérieure et sa charge utile en mode transport ; tandis qu'en mode tunnel, elle contient le paquet IP originel en entier (entête compris).

Le *trailer* Q-ESP inclut les champs suivants :*Padding, Pad Length* et *Next Header.* Ce dernier champ indique le protocole suivant (e.g., le protocole qui sera appliqué juste après Q-ESP). En mode transport, ce champ sera tout simplement égal au TLP, tandis qu'en mode tunnel, il sera égal à IP (puisque le protocole suivant dans ce cas s'occupera de la décapsulation IP).

Enfin, la zone d'authentification des données, dite ICV (pour, *Integrity Check Value*) est calculée à partir de l'entête Q-ESP et de la charge utile.

Concernant les champs restant de l'entête IP, nous considérons que la plupart des valeurs de ces

champs (*ToS*, *flags*, *fragmentation offset*, *TTL* et *checksum*) sont modifiables (peuvent être changés durant l'acheminement du paquet), ils sont par conséquent exclus du processus d'authentification. Cependant, contrairement à AH, nous pensons qu'authentifier les champs restants de l'entête IP (*version*, *header length*, *packet length*, *next protocol* et *ID*) n'a pas de sens puisque ces champs seront utilisés avant que le paquet atteigne la couche Q-ESP (*i.e.* avant la vérification de leur intégrité) ; ainsi, un changement de leur valeur n'affectera pas le traitement IPSec.

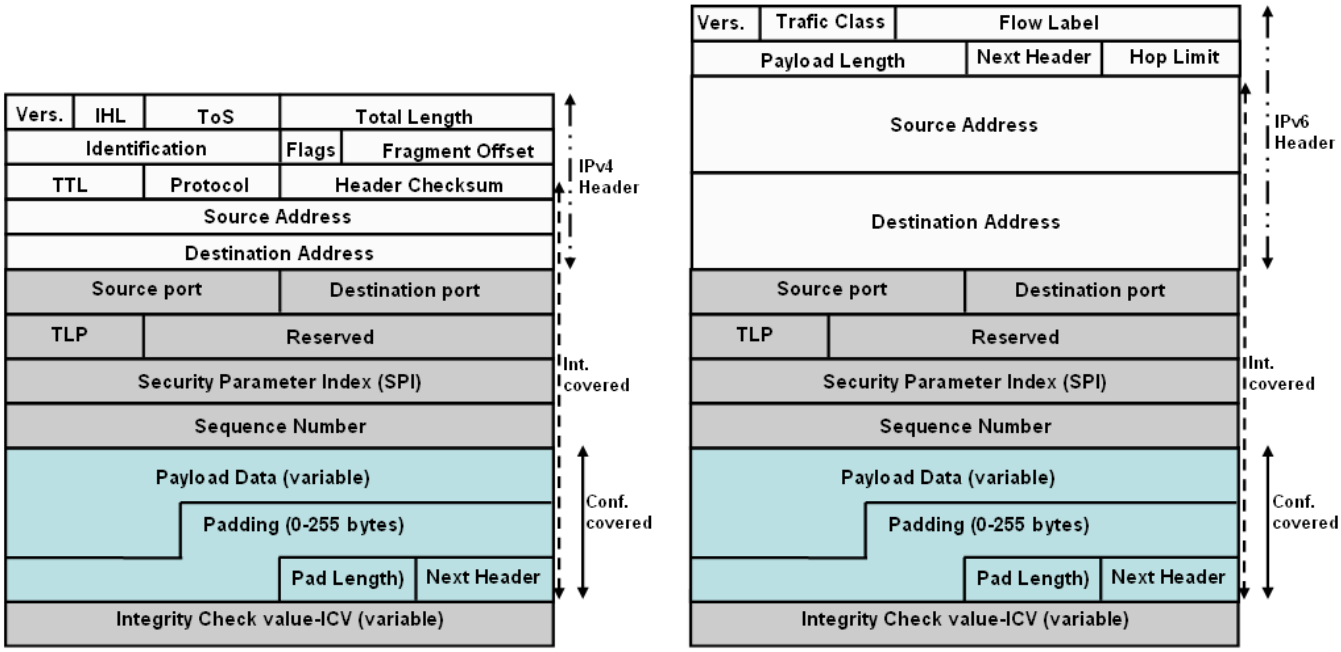


Figure 15 : Format d'un paquet Q-ESP en IPv4 et Ipv6.

B. Traitement Q-ESP

Dans la section précédente, nous avons décrit le format des entêtes de notre protocole. Il est maintenant temps de décrire comment l'émetteur construit le paquet Q-ESP à partir d'un paquet IP originel, et quelles sont les transformations cryptographiques qu'il subit ; et inversement, il faut détailler comment le récepteur traite le paquet afin d'effectuer les vérifications nécessaires et extraire la charge utile en clair. Les deux diagrammes de la Figure 16 illustrent les traitements sortants appliqués aux paquets IP (côté gauche de la figure) ainsi que les traitements que subissent les paquets entrants (côté droit).

Traitement des paquets sortants

O1 *: Recherche du SA*

Dès qu'un paquet IP est prêt à être émis, on cherche dans la base de données des politiques de sécurité (*Security Policy Database* ou SPD) pour déterminer si Q-ESP s'applique à ce paquet. Si oui, on cherche dans la base de données des associations de sécurité (*Security Association Database* ou SAD) pour récupérer l'association de sécurité (*Security Association* ou SA) à utiliser. Si celle-ci existe, on passe à l'étape suivante ; sinon on invoque sa création auprès du protocole *Internet Key Exchange* ou IKE qui sert à négocier dynamiquement les paramètres de la communication sécurisée [Harkins 1998].

Au terme de cette étape, nous obtenons une SA qui contient un index de paramètre de sécurité (*Security Parameter Index* ou SPI), un numéro de séquence initialisé, les algorithmes à utiliser pour le chiffrement et l'authentification, leurs clés secrètes ainsi que le mode (tunnel ou transport).

O2 *: Construction de l'entête IP*

Le traitement de cette étape dépend du mode de sécurité : en mode transport, l'ancien entête IP sera utilisé pour acheminer le paquet, tandis qu'en mode tunnel, nous construisons un nouvel entête IP pour encapsuler l'ancien entête IP. Pour construire le nouvel entête IP externe, les nouvelles adresses source et destination IP des passerelles seront placées dans l'entête IP externe. Les champs restants seront copiés depuis l'ancien entête IP vers leur correspondance dans le nouvel entête IP. Après cela, le champ TTL sera décrémenté. Notons que la valeur de la somme de contrôle (*checksum*) de l'entête sera calculée ultérieurement à l'étape O6.

O3*: Construction de l'entête Q-ESP*

Pour la construction de l'entête Q-ESP, nous effectuons les opérations suivantes:

- Copier les deux premiers champs (ports source et destination) de l'entête de la couche supérieure (TCP / UDP) et les mettre au début de l'entête Q-ESP.
- Copier la valeur du champ numéro de protocole dans l'entête IP (l'entête interne en mode tunnel) et l'affecter au nouveau champ TLP (*Transport Layer Protocol*) de l'entête Q-ESP.
- Mettre la valeur du numéro de protocole à 141 (le numéro de protocole proposé pour Q-ESP).
- Mettre le champ *Reserved* à zéro.
- Placer le champ SPI obtenu de la SA définie à l'étape O1, afin d'informer le récepteur sur la manière de traiter ce paquet.
- Incrémenter le numéro de séquence et le mettre à la dernière zone de l'entête.

L'entête Q-ESP est maintenant construit et contient les champs suivants : numéros des ports source et destination, TLP, champ réservé, SPI et numéro de séquence. Désormais, toutes les informations nécessaires pour effectuer la classification sont disponibles en clair au MF *packet classifier.*

O4 *: Chiffrement de la charge utile Q-ESP*

Afin de préparer la charge utile Q-ESP pour le chiffrement, on créé le *trailer*, puis on ajoute du bourrage en inscrivant la valeur de sa longueur dans le champ *pad length*. La valeur du *next header* et le contenu de la charge seront affectés par le mode de fonctionnement. En mode tunnel, l'ensemble du paquet IP original sera placé dans la charge utile et le *next header* sera fixé à 4 (encapsulation IP dans IP). En mode transport, la couche supérieure (entête et charge) sera mise dans la charge utile de Q-ESP et le *next header* contiendra la valeur du protocole de la couche transport (dans ce cas, on aura la même valeur que dans le champ TLP). La charge utile Q-ESP et le *trailer* sont chiffrés en utilisant des algorithmes tels que AES ou 3-DES conformément à l'équation (A1).

$$\mathbf{M_E} = \{\, \boldsymbol{Mp} \,||\, \boldsymbol{Tr} \,\}_{\boldsymbol{K_E}} \qquad \text{(A1)}$$

Où M_E est le message chiffré (*Encrypted*), les accolades illustrent l'opération de chiffrement, M_P est la charge utile (*Payload*) Q-ESP, T_r est le *trailer* Q-ESP, "|" est l'opérateur de concaténation, et enfin K_E est la clé de chiffrement.

O5 *: Calcul du contrôle d'intégrité*

Le calcul du contrôle d'intégrité (ICV) est réalisé en utilisant l'algorithme d'authentification standard spécifié par l'association de sécurité (SA), tel que SHA-2, conformément à l'équation (A2). La valeur

de l'ICV est ensuite inscrite dans le champ *Authentication Data Area*.

$$\mathbf{ICV = H(}\boldsymbol{M_H}\mathbf{||}\ \boldsymbol{M_E}\ \mathbf{||}\ \boldsymbol{Src\ IP}\ \mathbf{||}\ \boldsymbol{Dst\ IP}\mathbf{)}\boldsymbol{K_A} \qquad \text{(A2)}$$

Où *ICV* est la valeur du contrôle d'intégrité, *H* est l'algorithme de hachage par clé (type H-MAC), M_H (*header*) est l'entête Q-ESP, M_E est la charge utile (*Payload*) Q-ESP chiffrée (déjà calculée dans l'étape précédente selon l'équation A1), "*Src IP*" et "*Dest IP*" sont les adresses IP source et destination, et K_A est la clé d'authentification (utilisée par l'algorithme de hachage par clé).

O6 *: Finalisation*

La construction du paquet Q-ESP est complétée et la somme de contrôle (*checksum)* ICV de l'entête est placé à la fin du paquet Q-ESP.

Traitement des paquets entrants

I1*: Recherche du SA*

Dès la réception d'un paquet IP entrant contenant la valeur 141 dans le champ "numéro de protocole" (celui de Q-ESP) de son entête, on recherche dans la *SAD* en utilisant le *SPI* pour récupérer l'association de sécurité *SA* à utiliser. Si elle existe, on passe à l'étape suivante. Dans le cas contraire (le *SA* n'existe pas), on rejette le paquet et on en informe le protocole *IKE* afin de construire une nouvelle association de sécurité.

I2 *: Vérification du numéro de séquence*

Dans notre protocole, cette étape est obligatoire pour se prémunir des attaques par rejeu. Si le numéro de séquence du paquet est valide (i.e. Il n'est pas dupliqué et ne dépasse pas le numéro de séquence de la fenêtre contenu dans le *SA*), on passe à l'étape suivante. Sinon (i.e. numéro de séquence invalide), on rejette le paquet. Il est important de noter que la fenêtre ne doit pas avancer tant que le paquet qui doit provoquer sa progression n'a pas été authentifié.

I3 *: Vérification du contrôle d'intégrité*

La valeur du contrôle d'intégrité (ICV) est recalculée en utilisant l'algorithme d'authentification standard spécifié par le *SA* conformément à l'équation *A*2. L'*ICV* porte sur l'entête Q-ESP, sa charge utile, ainsi que les adresses IP source et destination de l'entête IP externe. Si les valeurs des *ICV* calculée et reçue sont égales, on passe à l'étape suivante. Sinon, on rejette le paquet.

I4 *: Déchiffrement de la charge utile Q-ESP*

On utilise l'algorithme standard de chiffrement (tels que AES) spécifié par l'association de sécurité *SA* et on déchiffre la charge utile Q-ESP à l'aide de la clé associée, conformément à (*A*3).

$$\mathbf{M_D = [}\ \boldsymbol{M_E}\mathbf{]}\boldsymbol{K_E} \qquad \text{(A3)}$$

Où M_D est le résultat du déchiffrement (la charge utile Q-ESP et son *trailer*), les crochets désignent la fonction de déchiffrement, M_E le message chiffré et K_E la clé de déchiffrement.

I5 *: Extraction du protocole encapsulé dans Q-ESP*

En mode transport, on extrait la PDU (*Protocol Data Unit*) TCP/UDP contenue dans la charge utile Q-ESP, tandis qu'en mode tunnel, on extrait l'entête interne du paquet IP encapsulé ainsi que sa charge utile, tous deux logés dans la charge utile de Q-ESP.

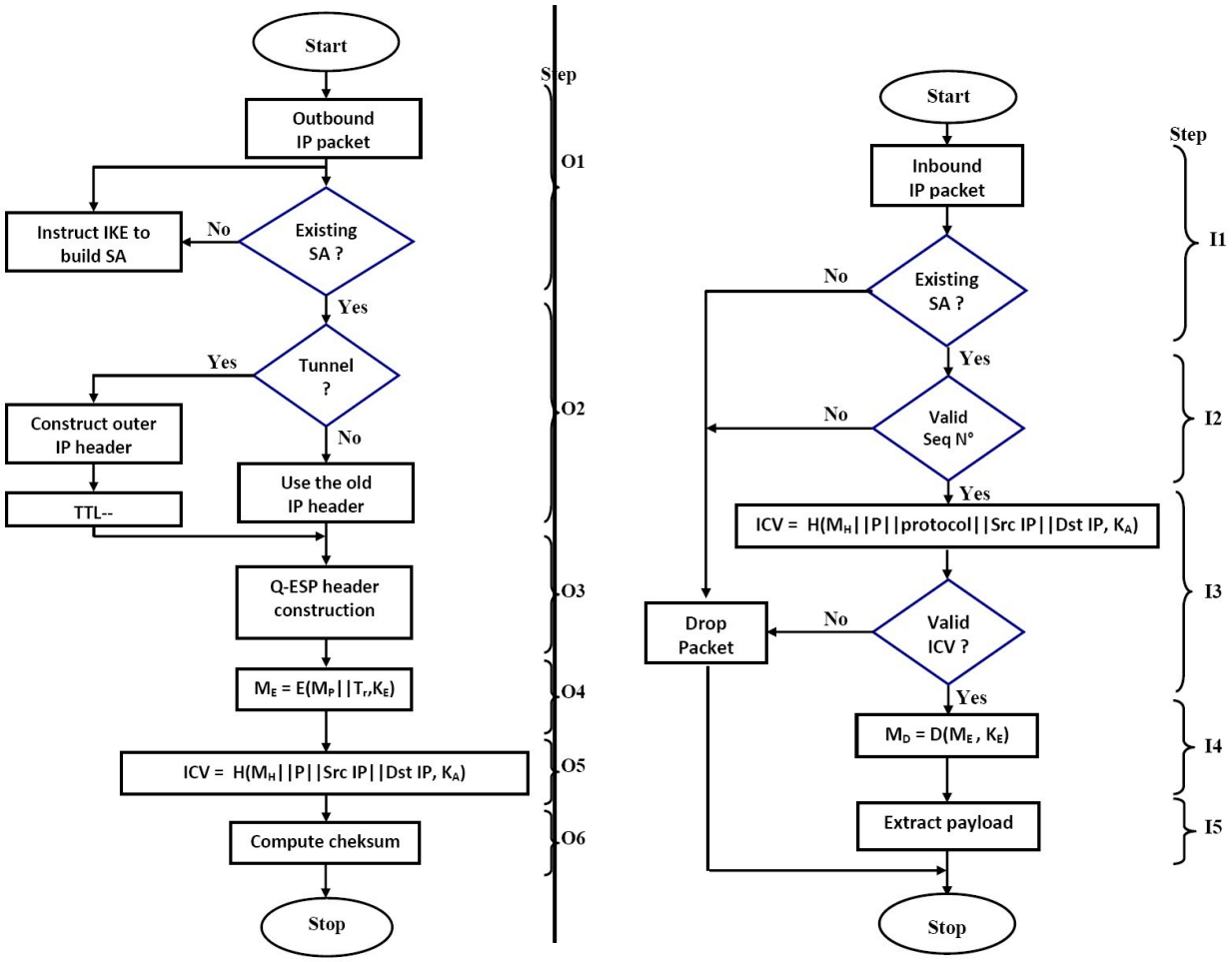


Figure 16. Traitement Q-ESP des paquets entrants (droite) et sortants (gauche).

C. Implémentation de Q-ESP

Notre implémentation inclut d'une part des correctifs du noyau (*patchs kernel*) pour NetBSD version 5 [NetBSD 2009], d'autre part des correctifs (*patchs*) *userland* pour l'outil *setkey* de configuration manuelle d'IPsec [Suominen 2004]. Nos patchs ont été apportés à l'implémentation *Fast IPSec* [Samuel 2003], celle-ci a été conçue pour de hautes performances (50% mieux que toute autre implémentation publique d'IPSec [Schneider 1994]) et bénéficie de l'utilisation du cadriciel cryptographique d'OpenBSD (*OpenBSD Cryptographic Framework*) [Keromytis 2003], migré par la suite vers *FreeBSD* et NetBSD. Ce cadriciel permet d'utiliser tout matériel de chiffrement pour accélérer le fonctionnement.

Nos patchs noyau (*kernel*) permettent notamment d'effectuer les procédés de transformations sur les paquets IP entrants et sortants tels que : la formation, l'insertion et l'extraction de l'entête Q-ESP, ainsi que les opérations d'authentification, de chiffrement et de déchiffrement. Ces patchs prennent également en compte la déclaration du protocole Q-ESP au sein de la pile IPSec sous le numéro de protocole 141.

La Figure 17 résume le traitement Q-ESP pour les paquets entrants et sortants dans le noyau NetBSD.

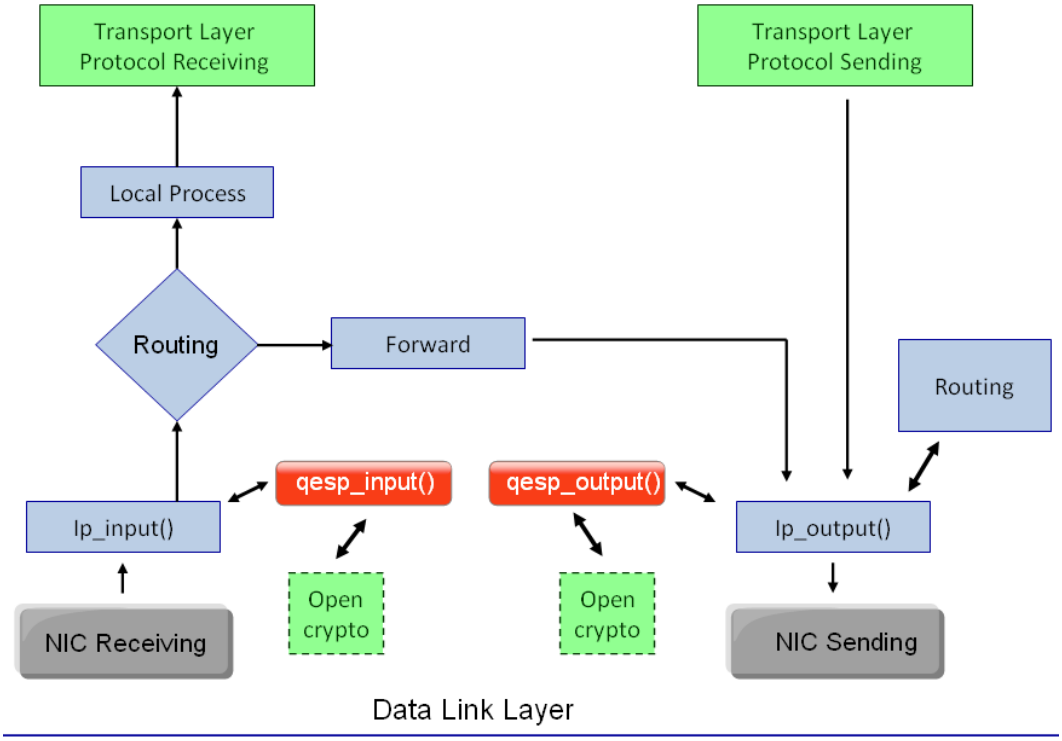


Figure 17. Traitement Q-ESP. Dans le noyau NetBSD.

Par ailleurs, nos patchs *Userland* permettent de modifier l'outil *Setkey* afin qu'il soit capable : (1) de configurer les associations de sécurité (SA) basées sur Q-ESP dans la SPD, et (2) de fixer la valeur des SPI des différentes entrées dans la SAD. Notons que ces deux tables sont stockées (et abritées) dans le noyau pour des raisons de sécurité. Parallèlement au développement de ces patchs, nous avons mis au point un *plug-in Wireshark* afin de simplifier les phases de débogage et de validation syntaxique du protocole. Avec ce *pluging*, *Wireshark* permet de visualiser clairement les échanges HTTP protégés avec Q-ESP. La Figure 18 en montre une capture de *Wireshark* reconnaissant notre protocole Q-ESP.

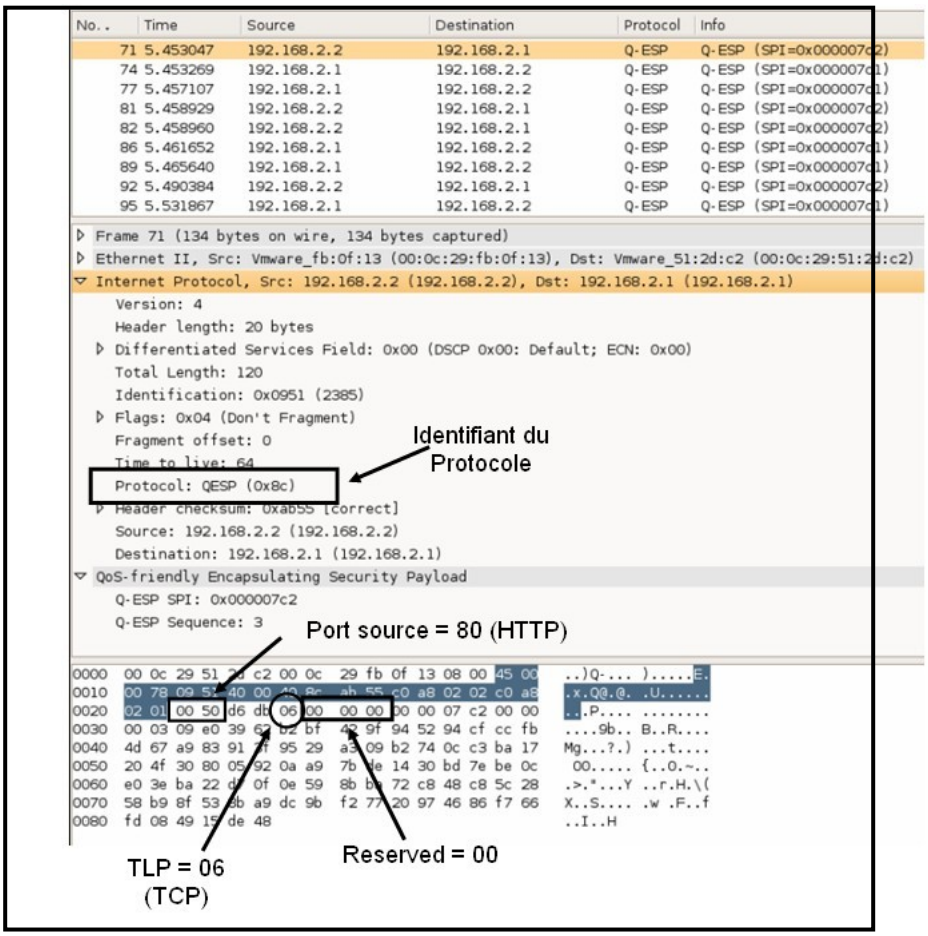


Figure 18 : Capture Wireshark montrant les flux HTTP protégés par Q-ESP en mode tunnel.

D. Évaluation des performances

Cette section vise à prouver que Q-ESP assure effectivement des besoins de sécurité et de QoS. Nous y présentons une partie de nos analyses incluant : (1) une étude comparative entre Q-ESP et IPSec, et (2) quelques une de nos expériences effectuées pour mesurer les métriques de QoS. Enfin, nous analysons les résultats obtenus.

Évaluation Analytique

En plus des services de sécurité assurés par IPSec, Q-ESP supporte la QoS en fournissant les informations nécessaires et suffisantes pour que les équipements de contrôle du réseau puissent réaliser un contrôle d'admission actif (processus connu sous le nom "*active admission control*"). De plus, il est clair qu'utiliser un protocole (qui fournit les services requis) au lieu de deux (AH et ESP) simplifie l'implémentation et l'administration d'IPSec. En outre, l'entête Q-ESP est de petite taille (16 octets), tandis que la somme des entêtes AH et ESP est plus grande (8+12=20 octets). Ce faisant, le surcoût (*overhead*) de Q-ESP est moindre que celui des protocoles AH et ESP utilisés conjointement.

En outre, Q-ESP empêche les attaques par rejeu (cette fonction est obligatoire dans Q-ESP, alors qu'elle est optionnelle dans ESP et AH). De la même manière, alors que l'authentification est

optionnelle pour ESP, elle est obligatoire pour Q-ESP, permettant ainsi de se prémunir contre les paquets malicieux avec des entêtes IP et Q-ESP valides, mais contenant une charge utile invalide (qui serait rejetée, mais seulement après l'application du processus de déchiffrement qui s'avère consommateur de ressources). Enfin, Q-ESP fournit l'authentification de l'origine des données (puisqu'il couvre l'essentiel des champs de l'entête externe IP en mode tunnel).

Concernant la complexité temporelle des opérations de traitement, Q-ESP suit globalement les mêmes étapes de traitement qu'AH et ESP ; seules deux nettes différences sont identifiables :

- Côté émetteur, Q-ESP copie les numéros de ports source et destination du protocole de transport de la couche supérieure dans son entête, le temps requis pour une telle opération est ainsi extrêmement faible, et peut être considéré comme négligeable.
- Étant obligatoire, l'authentification à travers une fonction de hachage avec clé (ICV) est d'abord réalisée par l'émetteur puis vérifiée par le récepteur.

Comme nous l'avons déjà mentionné, ESP authentifie seulement son entête et la charge utile protégée, il n'authentifie pas l'entête externe dans le mode tunnel, alors que dans ce même mode, AH authentifie le paquet entier, dont l'entête IP externe. D'un autre côté, l'authentification Q-ESP couvre son entête, la charge utile protégée, et seulement deux champs de l'entête IP externe. Cette différence (de la portée de l'authentification) peut affecter les performances. En considérant que nous utilisons l'algorithme SHA-1 pour le contrôle d'intégrité, et en se basant sur l'analyse de performance réalisée par [Elkeelany *et al.* 2002], la complexité temporelle des opérations réalisées par l'algorithme de contrôle SHA-1 est : $n * 1100$ *opérations*, où n est le nombre de blocs en entrée, donnée par :

$$n = ceiling\ (taille\ totale\ message / (taille\ du\ bloc = 512))$$

Dans la comparaison de Q-ESP avec ESP (Tableau 11), il semble intéressant de tenir compte des deux situations extrêmes : soit Q-ESP a le même nombre de blocs (n) qu'ESP, soit, dans le pire des cas, Q-ESP a (n+1) blocs (puisque son authentification couvre plus de champs). De la même manière, en comparant Q-ESP avec AH, nous pouvons considérer les deux mêmes situations extrêmes : soit AH a le même nombre de blocs (n) que Q-ESP, soit AH a (n+1) blocs, puisque son authentification couvre la totalité de l'entête externe du paquet IP. Il semble clair que les blocs supplémentaires coûteront une surcharge de traitement de 1100 opérations dans le cas de SHA-1 et 744 opérations dans le cas de MD5. Toutefois, soulignons que le nombre de blocs (n) dépend de la taille du paquet.

Tableau 11 : Q-ESP versus ESP and AH.

		ESP	**Q-ESP**	**AH**
Sécurité	Authentification	Optionnel	Automatique	Automatique
	Chiffrement	Automatique	Automatique	NA
	Anti-rejeu	Optionnel	Automatique	Optionnel
QoS	Surcoût	8 octets	16 octets	12 octets
	Contrôle d'admission	Non	Oui	Oui

Évaluation expérimentale

Afin d'évaluer les performances de Q-ESP, nous avons mis en place deux plate-formes Le premier type d'expérimentation vise à comparer le débit de traversée de Q-ESP avec celui d'ESP sans traitement QoS (c'est-à-dire en environnement *Best effort*). Le second type d'expérimentation a été effectué dans une situation de congestion ; le but étant de prouver que Q-ESP autorise bien le contrôle d'admission actif, et peut donc fournir un traitement différencié selon le type de flux et la politique de QoS adoptée.

Pour générer un trafic UDP, et donc simuler une application temps réel, nous avons choisi l'outil MGEN (*Multi-Generator*) [MGEN 2009] ; tandis que pour les trafics TCP, nous avons utilisé l'outil *Iperf* [Iperf 2009].

Pour le premier test, nous avons mis en place le banc de tests illustré à la Figure 19 et nous avons mesuré le débit de traversée (*throughput*).

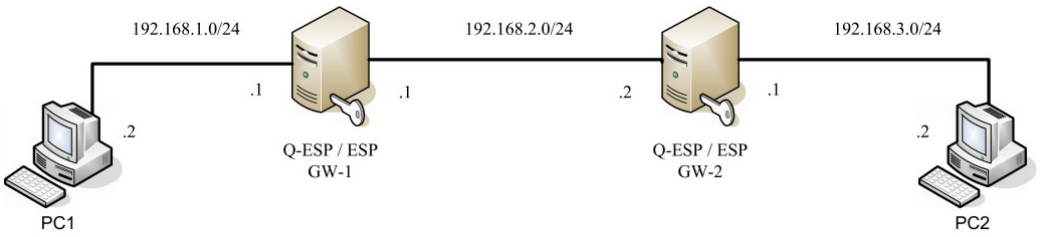


Figure 19 : Plate-forme de mesure de temps de traitement.

Les deux passerelles de sécurité (GW-1 et GW-2) sont connectées avec une liaison point à point. Les protocoles Q-ESP et ESP sont installés respectivement dans les deux passerelles de sécurité. Deux scénarios ont été mis en place : le premier utilise des messages ICMP (*ping*) pour mesurer le temps aller-retour (RTT), et le second utilise l'outil MGEN pour mesurer le débit de traversée.

Scénario #1

Nous avons généré des messages *ping* avec des paquets de différentes tailles (64, 128, 256, 1024 et 2048 octets), et en utilisant différents algorithmes de chiffrement/authentification. Les résultats sont présentés dans la Figure 20.

Ces mesures montrent que les RTT mesurés pour ESP et Q-ESP sont presque identiques, ce qui signifie que ces deux protocoles ont pratiquement les mêmes temps de traitement. En réalité, ce temps dépend de deux facteurs : la taille du paquet et l'algorithme (de chiffrement ou authentification) utilisé. Il est évident que lorsque la taille du paquet augmente le temps de traitement augmente. Concernant l'algorithme utilisé, nous pouvons constater que l'algorithme 3DES a le plus gourmand en terme de temps de traitement.

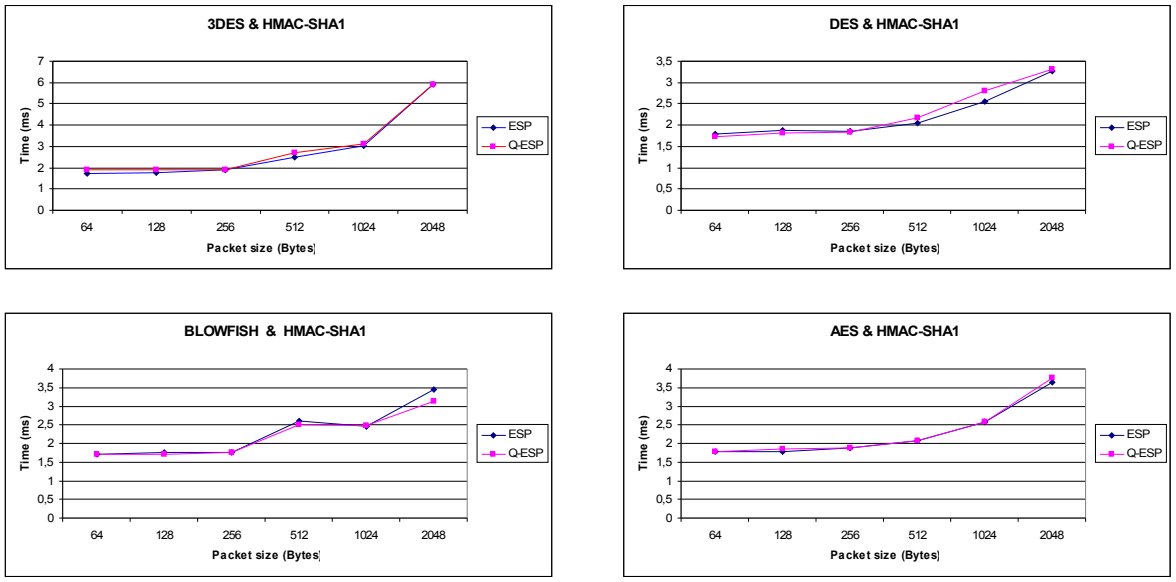


Figure 20 : Mesures du RTT pour ESP et Q-ESP en utilisant différents algorithmes.

Scénario #2

Dans le deuxième scénario, MGEN est installé sur les postes réalisant la mesure (PC1 et PC2). MGEN est utilisé pour envoyer des paquets UDP de PC-1 à PC-2 et mesurer le débit de traversée. La passerelle Gw-1 prend en charge le traitement en émission, tandis que Gw-2 prend en charge le traitement en réception. La Figure 21, montre les résultats obtenus.

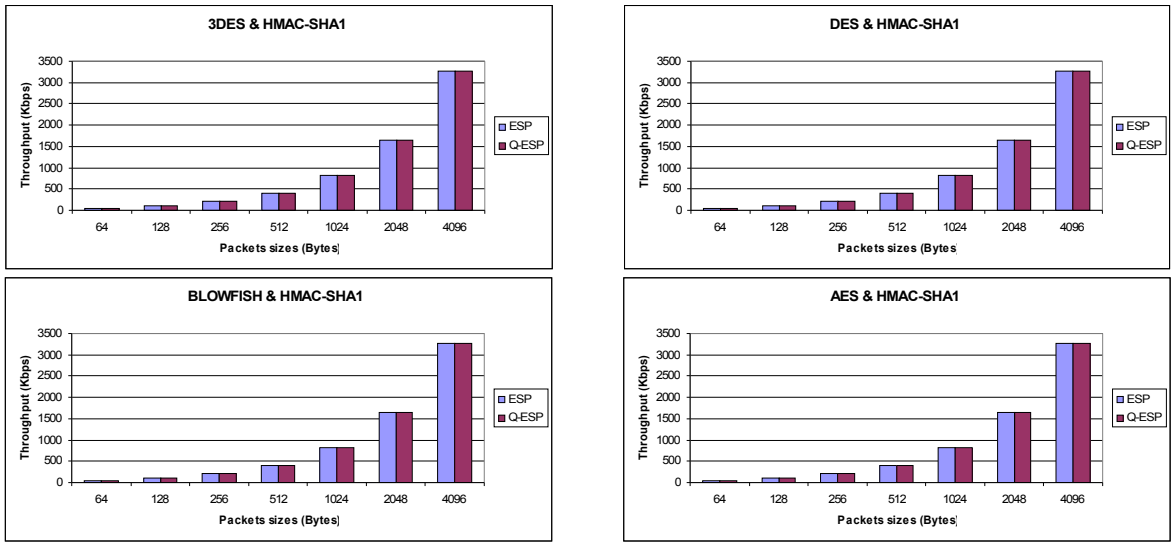


Figure 21 : Débit de traversée (*Throughput*) pour ESP et Q-ESP en utilisant différents algorithmes.

Les graphes de la Figure 20 montrent qu'ESP et Q-ESP ont presque le même débit de traversée quelque soit l'algorithme de chiffrement/authentification utilisé. Toutefois, le protocole Q-ESP affiche

une capacité de transfert relativement faible par rapport à ESP, validant ainsi nos conclusions théoriques. Ceci est principalement dû à l'ajout des numéros de port source, port destination et TLP au niveau de l'en-tête Q-ESP, entraînant une très légère augmentation du débit protocolaire et une toute petite réduction de la capacité de transfert.

A partir de ces résultats, nous pouvons conclure que dans un environnement *Best Effort*, on peut utiliser Q-ESP au lieu d'ESP sans vraiment affecter les performances en terme de temps de calcul.

Passons maintenant aux situations qui nous intéressent le plus, à savoir celle où la QoS est importante. Nous avons ainsi comparé Q-ESP et ESP par rapport aux métriques de QoS : débit, délai, et taux de perte. La figure 22 montre la configuration du banc d'essai utilisé.

La configuration de la plate-forme se compose de deux domaines reliés par un simple nuage *DiffServ*. Chaque domaine contient trois PCs (PC 2 à 4) pour créer des situations de congestion, et un PC (PC1) qui génère le flux à protéger. Ainsi, intégrant les protocoles Q-ESP et ESP, les passerelles ne protègeront que le trafic généré par PC1 en fournissant les services de chiffrement, d'authentification et d'anti-rejeu.

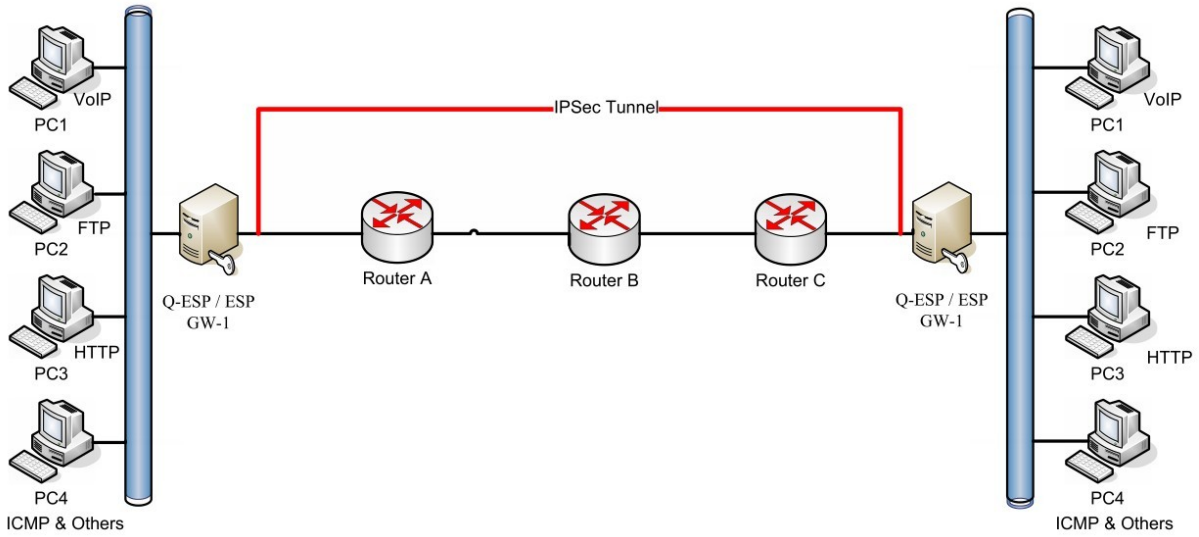


Figure 22 : Plate-forme de test de la fonction de marquage.

Le nuage DiffServ contient deux routeurs de bordure (*Edge routers*) pour effectuer les opérations de contrôle d'admission actif (telles que la classification et le marquage des paquets), et un routeur de cœur (*core router*) pour gérer différemment les trafics en fonction de leurs priorités.

Pour effectuer les tests, MGEN a été utilisé dans PC1 pour générer des paquets UDP et simuler ainsi un flux temps réel (VoIP). Plus précisément, pour simuler les communications simultanées, nous avons généré des flux avec des fréquences différentes (100, 200, ..., 1000 paquets par seconde). Ces paquets ont été protégés par les passerelles Q-ESP/ ESP. Les autres PCs ont été utilisés pour produire différents types de flux (FTP, HTTP, UDP et ICMP) avec différentes tailles de paquets. Ces flux sont destinés à simuler le trafic sur le réseau réel.

Nous avons mis en place deux scénarios. Dans le premier scénario, nous avons mesuré les

performances de Q-ESP et ESP en cas de congestion du réseau, mais sans activer le traitement QoS au niveau de Q-ESP. Dans le deuxième scénario, les mêmes expériences ont été réalisées mais en activant le traitement QoS dans Q-ESP. Plus précisément, les routeurs de bordure *(Edge routers)* sont configurés pour traiter le trafic Q-ESP avec une haute priorité et les trois autres trafics avec une faible priorité. Nous avons effectué ces tests en utilisant l'algorithme AES pour le chiffrement et l'algorithme HMAC-SHA1 pour l'authentification et le contrôle d'intégrité. Les figures 23, 24 et 25 présentent les résultats obtenus.

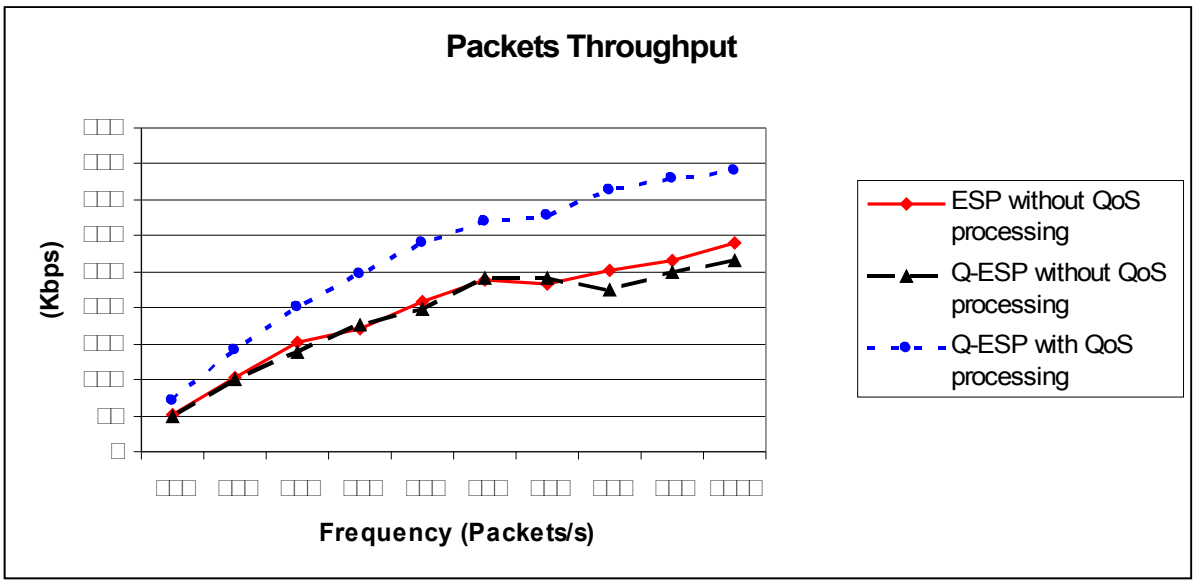


Figure 23 : Mesures de débit (*Throughput*).

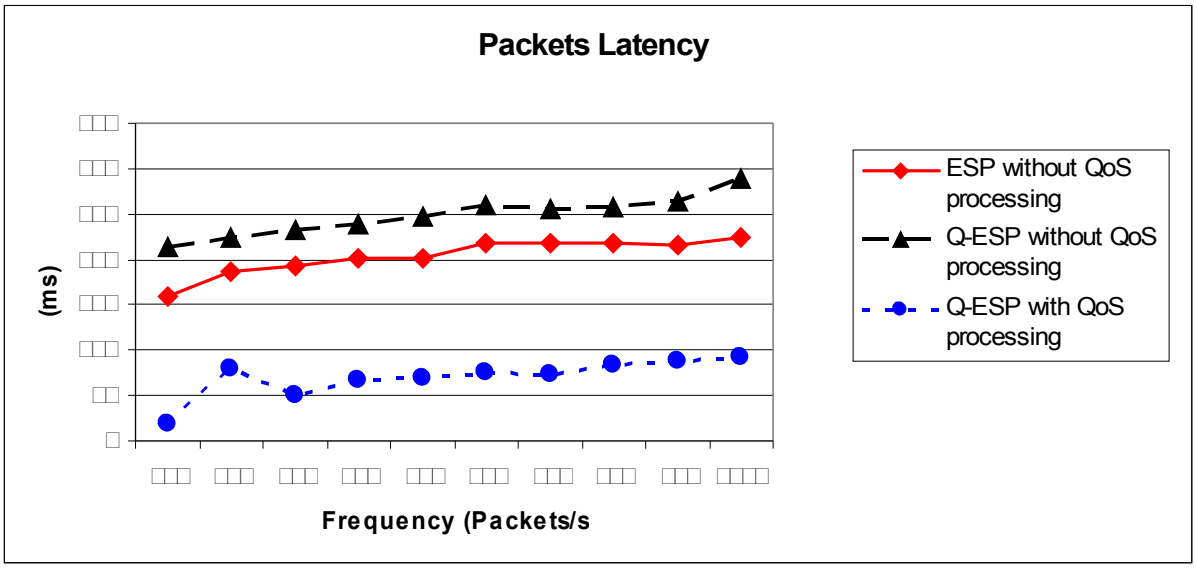


Figure 24 : Mesures du délai (*Latency*).

Le débit est le premier paramètre important que nous avons mesuré. Comme le montre la Figure 23, "Q-ESP avec traitement QoS activé" a un débit plus élevée en comparaison à ESP et à Q-ESP sans QoS.

Concernant le délai (latence), comme le stipule la référence [Szigeti & Hattingh 2004], ses valeurs ne doivent pas dépasser 150 ms pour les trafics temps réel. Malheureusement, dans le premier scénario (sans traitement QoS), la latence des flux Q-ESP et ESP dépasse largement cette limite et montent en effet vers la valeur 300 ms (Figure 24). Nous attribuons ce résultat à la forte charge du réseau. En effet, comme les paquets Q-ESP et ESP sont traités d'une façon *Best-Effort*, ils sont servis comme les autres trafics (sans préférence), ce qui augmente la concurrence pour l'accès au médium de communication avec les autres trafics. Ainsi, dans des situations de congestion du réseau, les paquets (ESP et Q-ESP sans activation de la QoS) sont stockés dans les mémoires des routeurs en attendant qu'ils soient traités pour accéder au médium. Il est clair que ce temps d'attente est la principale source de retard.

Dans le deuxième test, le traitement QoS est activé pour les paquets Q-ESP. Le routeur de bordure (*Edge routers*) identifie les paquets Q-ESP, puis les classe et les marque avec la plus haute priorité. De cette manière, le trafic Q-ESP est servi plus rapidement que les autres trafics, et donc, sa latence est très faible et ne dépasse pas 100 ms comme le montre la Figure 24.

Enfin, le dernier paramètre QoS que nous avons considéré est le taux de perte. Comme le montre la Figure 25, on peut constater qu'il existe une différence importante entre le pourcentage de perte de paquets dans les deux scénarios. Dans le premier scénario (Q-ESP et ESP sans traitement QoS), le pourcentage de perte de paquets est plus élevé que le deuxième test où Q-ESP intègre le traitement QoS. Ceci est attendu, étant donné que ESP et "Q-ESP sans traitement QoS" partagent la même file d'attente avec les autres trafics, ce qui augmente leur probabilité de suppression. Heureusement, quand le traitement QoS est activé, le trafic Q-ESP est attribué à une file d'attente différente, ce qui diminue notablement sa probabilité de suppression. Dans ce cas, comme le montre Figure 25, « Q-ESP avec traitement QoS activé » a de très faibles taux de pertes, même en situations de congestion réseau.

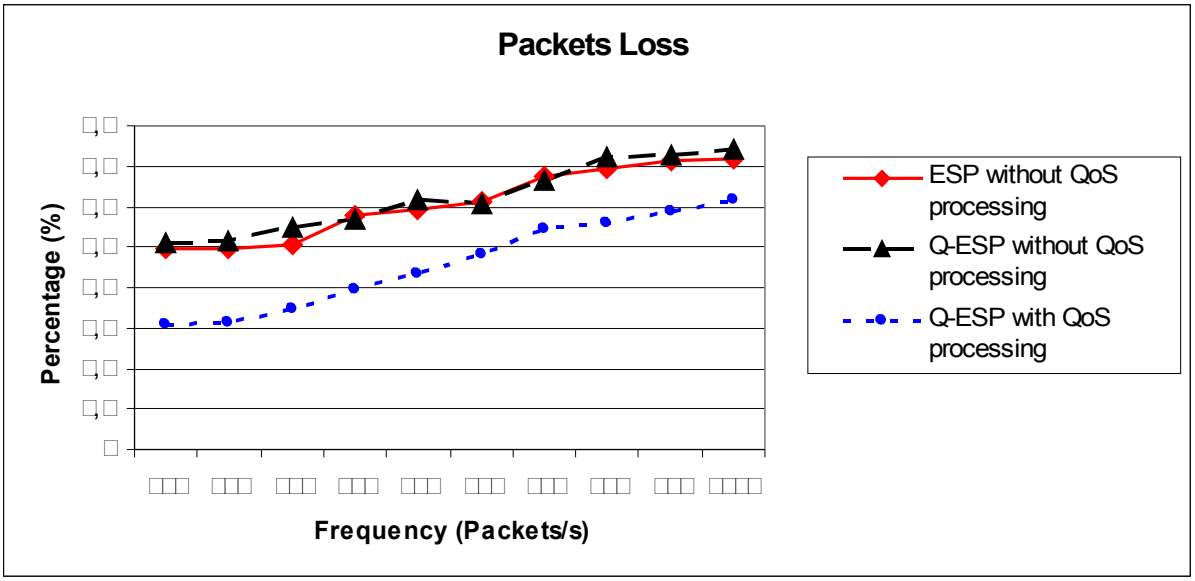


Figure 25 : Mesures du taux de perte.

D’après nos études analytiques et expérimentales, nous pouvons affirmer que Q-ESP a pratiquement le même débit qu’ESP dans un environnement *Best-Effort*, tandis que dans environnement QoS, Q-ESP a de meilleures performances en terme de QoS, toute métrique confondue. Q-ESP est donc une solution puissante pour les environnements où les contraintes temporelles sont importantes, notamment les applications temps réel et les réseaux avionique. Q-ESP permet effectivement aux éléments actifs du réseau de classer les paquets et d'effectuer le contrôle d'admission actif. En outre, Q-ESP offre une protection efficace en matière de sécurité étant donné qu'il fournit à la fois les services de sécurité d'AH et d'ESP.

IV. Conclusion

Dans ce chapitre, nous avons présenté un nouveau protocole de sécurité appelé Q-ESP, celui-ci associe les fonctions de sécurité offertes par les protocoles IPSec ESP et AH, tout en prenant en charge la QoS en permettant le contrôle du marquage des flux. Q-ESP fournit ainsi des fonctionnalités qui font de lui un candidat idéal pour remplacer AH et ESP. Du point de vue de la sécurité, utiliser un protocole unique et simple qui fournit toutes les fonctionnalités (de sécurité) requises au lieu de deux protocoles (AH et ESP), simplifie à la fois l'implémentation et l'administration des tunnels sécurisés.

Ce travail s'est essentiellement déroulé dans le cadre de la thèse de M. Mostafa que je co-encdare et qui a démarré en septembre 2007. Il a donné lieu à plusieurs publications notamment un article de journal [Abou El Kalam *et al.*, 2009b], trois conférences internationales [Mostafa *et al.* 2008a, Mostafa *et al.* 2008a, [Mostafa *et al.* 2009] et trois rapports de contrat [Abou El Kalam 2008c, [Abou El Kalam 2008d, [Abou El Kalam 2009]. Il sera notamment étendu, dans le cadre de la thèse de Warodom qui a démarré en septembre 2009, aux réseaux nouvelles génération (NGN) afin d'y développer de nouveaux protocoles assurant à la fois la sécurité et la QoS.

Par ailleurs, dans une perspective de standardisation, nous avons tenu lors de la spécification de Q-ESP à respecter certains critères de base, notamment :

- Extensibilité et évolutivité permises notamment grâce aux champs réservés au niveau de l'entête de Q-ESP.
- L'attribution d'un nouveau numéro de protocole, même si celui-ci doit être adopté par l'IANA dans une étape ultérieure du processus de normalisation.
- L’existence du champ *Next Header* permet d’utiliser Q-ESP avec d’autres protocoles.
- La différence par rapport au protocole existant ESP reste acceptable et n'induit pas un changement majeur au niveau des équipements réseau.

D'ailleurs, afin de mener à terme nos travaux, un draft IETF est rédigé à cet effet et des contacts ont été entrepris avec Cisco et Sun microsystems.

Chapitre 4 : Évaluation des outils de sécurité

I. Motivation

Dans le premier chapitre nous nous sommes intéressés à l'expression des politiques de sécurité pour les infrastructures critiques en particulier et les systèmes hétérogènes, multi-organisationnels, dynamiques et distribués en général. Ce travail, a débouché sur le modèle de contrôle d'accès PolyOrBAC. Ensuite, nous avons expliqué qu'il y a actuellement un fort besoin d'exprimer (au niveau de la politique de sécurité) non seulement des permissions, interdictions et obligations, mais aussi des recommandations. Nous avons ainsi proposé notre langage de spécification des recommandations et nous avons présenté sa syntaxe, sémantique, règles d'inférences ainsi que son axiomatique.

Néanmoins, avec ces modalités d'accès, il n'est pas impossible de dériver d'une part, des règles qui interdisent ou déconseillent une certaine action, et d'autres part des règles qui autorisent, obligent ou recommandent ces mêmes actions pour les mêmes utilisateurs. Pour résoudre ce type de problèmes, détecter et résoudre les conflits éventuels de modalités, nous avons proposé, dans le deuxième chapitre, l'utilisation de la programmation logique par contrainte (PLC).

Une fois spécifiée et vérifiée, une politique de sécurité doit être implémentée par des mécanismes appropriés de contrôle d'accès. Par exemple, avant de mettre en place un VPN, il faut répondre à des questions du type : quelles sont les parties qui doivent, peuvent, ne peuvent pas ou ne doivent pas communiquer ? Quelle information doit rester confidentielle et vis-à-vis de qui ? Quelle information doit être authentifiée et par qui ? Et en conséquence, à qui distribuer les clés de déchiffrement et d'authentification ? Etc. Ainsi, avoir une politique de sécurité est un préalable essentiel à la mise en place de mécanismes de sécurité tels que les VPN.

Néanmoins, l'implémentation et le déploiement d'une politique de sécurité peut introduire des contraintes de fonctionnement qui n'ont pas été identifiées dans la phase de spécification de la politique. Par exemple, introduire un surcoût et une surcharge de traitement qui n'est malheureusement pas acceptable dans les systèmes temps réel en général et les réseaux avioniques embarqués en particulier. Pire encore, le déploiement d'une politique de sécurité peut conduire à cacher certaines données (pour des raisons de confidentialité) alors que ces données sont essentielles pour la mise en œuvre de la QoS, autre besoin fort des réseaux à contraintes temporelles. Dans ce cas, le déploiement de la politique de sécurité se retrouve en conflit avec les mécanismes de QoS. Pour pallier ce type de problèmes et satisfaire à la fois des besoins de sécurité et de QoS, nous avons amélioré le protocole de sécurité IPSec pour la prise en compte de la QoS. Aussi, avons nous proposé dans le troisième chapitre le protocole Q-ESP : sa spécification, son fonctionnement, son implémentation ainsi que son évaluation.

Néanmoins, compte tenu de contraintes liées au fonctionnement du système ou à d'autres politiques (e.g., de QoS), la politique de sécurité peut être contournable, mal implémentée (sa conception est non-

conforme à la vie opérationnelle) ou tout simplement incomplète (ne couvre pas tous les accès possibles, conséquence d'une faute de conception ou d'un choix fonctionnel délibéré). Ainsi, en plus des outils et mécanismes préventifs pour faire appliquer une politique de sécurité, il convient souvent de renforcer la sécurité par d'autres contre-mesures, tels que les systèmes de détection d'intrusion (IDS, pour *Intrusion Detection System*). De manière globale, un IDS peut être défini comme l'implémentation des pratiques et mécanismes qui participent au diagnostic des attaques, ou à la détection des erreurs qui peuvent mener à la défaillance d'un système de sécurité ou à la violation des politiques de sécurité.

Malgré l'importance de tels outils, l'expérience montre qu'ils sont malheureusement peu fiables, car basés souvent sur des données empiriques (par exemple, des signatures d'attaques) ou sur des paramètres relatifs au symptôme d'attaques et non à leurs effets. Toutefois, ces données ne reflètent généralement que la connaissance que l'on possède du système ou des attaques potentielles. Une alerte levée par l'IDS ne correspond donc pas systématiquement à une violation effective de la politique, mais seulement de la présence d'un symptôme, associé empiriquement au risque de violation de la politique. En effet, face à l'augmentation du nombre et à la complexité des attaques, les systèmes de détection d'intrusions souffrent de taux élevés de fausses alertes ou de faux négatifs (attaques non détectées), ce qui les rend inefficaces, voir parfois inutiles. Des moyens de test et d'évaluation de tels outils s'imposent ainsi, notamment pour déterminer la qualité de détection des IDS et de leurs algorithmes de détection. C'est dans cette problématique que s'inscrit le présent chapitre.

Nous commençons par présenter notre méthodologie d'évaluation. Ensuite, afin d'obtenir des données de test représentatives, nous décrivons notre classification des attaques ainsi que notre modèle générique de processus d'attaques qui met en évidence la dynamique des activités d'attaques. Ce modèle a l'ambition de générer un nombre important de scénarios d'attaques qui soient le plus possible représentatifs et variés. Enfin, nous présentons notre outil de génération et d'injection des attaques et nous analysons les résultats de nos tests appliqués à différents IDS avec différentes configurations.

II. Notre méthodologie

Nous distinguons deux grandes classes d'évaluation des IDS : évaluation analytique et évaluation par test. Généralement, la première technique se base sur une modélisation du système étudié, et peut être appliquée à toute étape du cycle de développement ; tandis que la deuxième injecte des données réelles à une implémentation (un prototype, par exemple) du système sous test [Abou El Kalam & Gad 2006a, Gad & Abou El Kalam 2006].

Nombreuses sont les évaluations existantes [Bishop 1999, Kumar 1995, Hansmann 2003, Lindqvist 1997, Weber 1998, Kendall 1999, Lippmann *et al.* 2000a, Lippmann *et al.* 2000b, Howard 1998, Alessandri04]. Une analyse et discussion de ces travaux est disponible dans [Abou El Kalam & Gad 2006a, Gad 2008]. De manière globale, nous identifions plusieurs faiblesses dans les méthodes classiques d'évaluation. Le premier point à noter est l'*utilisation d'approches non systématiques* [Gad & Abou El Kalam 2006]. En effet, la plupart des méthodes de test des IDS sont des approches plutôt *ad hoc* : la sélection des paramètres du système, des facteurs, des métriques et des données de test est souvent arbitraire et non justifiée.

La deuxième critique est la *non-représentativité* des données de test. Ni le trafic de fond, ni les données d'attaques ne correspondent à la réalité d'Internet. L'IDS évalué se comporte ainsi

différemment sous test et quand il est déployé dans un environnement réel. Si on prend, par exemple, les données de test de la DARPA [Lippmann *et al.* 2000b], le trafic généré (de quelques Kbits/s) est considérablement plus faible que celui attendu (plusieurs Mbits/s). De plus, les données de test (trafic de fond) ont été basées sur des statistiques prises dans différents réseaux, mais leurs caractéristiques, en particulier celles liées à la génération de fausses alertes, n'ont pas été validées.

La troisième faiblesse que nous soulevons est la *non-pertinence des métriques*. On a souvent tendance à sélectionner des métriques faciles à mesurer, sans voir l'étendue réelle de leur efficacité. Parmi les métriques qui sont souvent utilisées dans l'évaluation des IDS, on peut citer le taux de détection, le rapport de détection (*detection ratio*), le taux de fausses alertes et le rapport de fausses alertes (*false alarm ratio*). Ces métriques ne sont pas toujours adéquates ou tout au moins, pas toujours justifiées. Si on prend le taux de fausses alertes comme exemple, il peut avoir plusieurs définitions selon la nature du dénominateur. Ce taux peut en effet être défini comme le nombre de fausses alertes divisé par le nombre total d'alertes, ou par le nombre de sessions, ou par le nombre de paquets. Néanmoins, même si le taux de fausses alertes par paquet peut avoir un sens pour un IDS qui applique une simple analyse de chaque paquet, il n'en est pas de même pour un IDS qui analyse des sessions et contrôle l'état de connexion.

Afin de pallier l'ensemble de ces limitations, il faut tout d'abord définir une méthodologie systématique d'évaluation des IDS [Abou El Kalam & Gad 2006b]. À cet égard, nous partons du constat que pour arriver aux résultats escomptés, il faut combiner "évaluation analytique" et "évaluation par test", ce qui est absent dans pratiquement toutes les évaluations existantes des IDS. La première technique aide à comprendre, spécifier et modéliser les éléments les plus importants (IDS, environnement, attaques, etc.), tandis que la deuxième tient compte de ces informations pour lancer les tests de manières réfléchie et bien ciblée. En effet, il serait illusoire de croire que les résultats de l'évaluation sont fiables si on se contente de lancer des tests choisis plus ou moins arbitrairement sans tenir compte des modèles d'attaques et des processus d'attaques. Et inversement, se contenter d'une évaluation analytique ne conduirait pas à des résultats fiables, car compte tenu de la complexité du domaine, toute spécification (d'Internet, des IDS, des attaques, etc.) sera nécessairement limitée ou même parfois erronée. Avec les résultats d'une évaluation par test, on peut par contre revenir à la spécification pour comprendre ce qui fonctionne et ce qui est défaillant, puis corriger d'éventuelles erreurs et limites dans la spécification.

Dans notre méthodologie, nous distinguons deux grandes phases [Gad & Abou El Kalam 2006] :

– *Phase de préparation* : celle-ci commence par identifier les besoins de l'utilisateur final (taux de détection, réactivité, facilité de mise en œuvre, adaptabilité, performances, diversité des canaux d'alerte, etc.). Ensuite il conviendrait de spécifier les caractéristiques de l'environnement et du contexte de test (dans lequel l'IDS sera déployé) ainsi que celles de l'IDS cible de l'évaluation. Les facteurs (paramètres contrôlables et ajustables par l'évaluateur) du système et des données de test sont également spécifiés. Par exemple, la bande passante et la taille des paquets du trafic de fond sont des facteurs plutôt liés aux données de test des NIDS (pour *Network Intrusion Detection System*). Bien évidemment, les facteurs ainsi que la technique d'évaluation ont un impact sur la sélection des métriques et des données de test.

– *Phase d'expérimentation* : consiste à préparer et exécuter les expériences, prendre les mesures, calculer, analyser, présenter et enfin, interpréter les résultats.

Détaillons maintenant quelques-unes des étapes de la phase préparatoire. Tout d'abord, l'identification des *besoins de l'utilisateur final* est importante dans la mesure où les facteurs, métriques et méthodes qui seront utilisés pour une évaluation orientée performance par exemple, seront naturellement différents de ceux choisis lorsqu'on s'intéresse plutôt à la robustesse à certains types d'attaques (dans ce deuxième cas, on s'intéresserait plutôt à mesurer des métriques telles que le taux de faux négatifs).

De la même manière, l'*environnement* peut différer d'une évaluation à une autre. Par exemple, les caractéristiques d'un réseau académique sont différentes de celle d'un réseau militaire, elles-mêmes différentes d'un environnement commercial. Des connaissances sur le système d'exploitation, les serveurs, les plate formes, les fichiers ainsi que les bases de données spécifiques à chaque environnement peuvent être pertinentes pour la procédure d'évaluation ; par exemple, pour aider à choisir le type de trafic de fond, les attaques à considérer ou à injecter à l'IDS sous test.

Étant donné qu'il est pratiquement impossible de calculer tous les aspects liés à l'environnement, les évaluateurs ne peuvent ajuster et modifier que les paramètres contrôlables, nommés ici *facteurs*. Parmi l'ensemble des facteurs, il faut en identifier ceux qui vont nous aider à comprendre le système évalué. Par exemple, pour les NIDS, ces facteurs peuvent être : la composition du trafic (type de trafic, longueur des paquets, contenu de la charge utile, utilisation de la bande passante) ; tandis que pour les HIDS (Pour *Host-based Intrusion Detection System*), on peut identifier des facteurs comme le système (plate-forme, version, …) où l'IDS est déployé, les applications et services qui tournent sur la machine, les vulnérabilités non encore corrigées, etc. On peut également considérer des facteurs spécifiques à l'IDS lui-même, notamment les règles de signatures, l'algorithme de détection, etc. Le type d'algorithme d'apprentissage ainsi que le seuil du profil sont deux exemples de facteurs associés aux IDS basés sur la détection d'anomalie.

Le *choix et la construction des données de test* (activités normales et intrusives) sont également des étapes essentielles. En effet, l'évaluation sera biaisée si les données de test choisies ne sont pas représentatives (de toutes les attaques possibles et éventuelles) ou ne correspondent pas aux besoins de l'utilisateur ou à l'environnement dans lequel l'IDS est (ou sera) déployé.

Enfin, rappelons que le choix des *métriques* est très important dans le processus d'évaluation ; en effet, si elles sont mal définies, les résultats de l'évaluation peuvent être faux ou biaisés.

III. Multi-Modèle pour la génération des données de test

La génération de données représentatives de test est sans doute l'un des problèmes les plus important lors de l'évaluation des IDS. La plupart des évaluations par test existantes étaient plus ou moins défaillantes car justement, elles passent directement à la phase d'expérimentation sans trop tarder sur la manière de choisir et générer les données de test. Dans notre vision, il faut tout d'abord caractériser le trafic réel, identifier les cas de test les plus pertinents et spécifier les données de test avant de passer à la génération de telles données. Rappelons que les données de test sont de deux types : des *données de fond* qui correspondent à des activités normales sans risque (ex. trafic réseau) et des *données d'attaque*s (ayant des fins malveillantes) qui correspondent à des actions exécutées par les attaquants.

En ce qui concerne les *données de fond*, elles doivent être réalistes et doivent refléter un trafic Internet "sain". Deux solutions existent : rejouer du trafic existant ou générer un trafic synthétique.

Rejouer un trafic existant serait dangereux dans la mesure où on ne peut totalement garantir que ce trafic ne contient pas des flux "illégaux" ou malveillants. À l'inverse, si on utilise un générateur de trafic, on peut être certain du trafic qu'on injecte, mais le problème reste la représentativité des données. En effet, nous avons testé plusieurs générateurs de trafic, notamment M-GEN, Ipef, NT Gen, GenSyn, TfGen, mais malheureusement, ils présentent des limites dans la mesure où certains ne gèrent pas des sessions impliquant différents utilisateurs (plusieurs utilisateurs qui envoient des paquets à plusieurs destinations à l'intérieur ou à l'extérieur du réseau local), d'autres ne gèrent pas les flux multi-services, etc. Dans nos tests, nous avons finalement opté pour étendre D-ITG afin de couvrir toutes les fonctionnalités dont nous avons besoin [Gad *et al.* 2009].

En ce qui concerne les *données malveillantes*, l'augmentation significative du nombre et de la complexité des attaques pose de sérieux problèmes pour les évaluateurs des IDS. En effet, comment tester efficacement l'IDS et avoir la certitude que celui-ci se comporte correctement (par exemple, génération d'une alarme pour toute tentative d'intrusion, pas de fausse alarme, etc.) pour toutes les attaques existantes voir inconnues ? Puisqu'il est impossible de tester les IDS vis-à-vis de toutes les attaques, il est indispensable de trouver une manière de sélectionner un ensemble de cas de tests pertinents et représentatifs. Pour répondre à ce besoin, nous nous sommes inspirés du concept de *classe d'équivalence,* bien connu dans le domaine du test de logiciel. Il consiste à réduire considérablement les cas possibles en construisant des classes d'attaques. On part donc du principe que n'importe quelle instance d'attaque d'une classe donnée produira les mêmes effets, et donc génèrera les mêmes résultats.

Or, en général, afin d'arriver à ses fins, un attaquant ne lance jamais une attaque élémentaire, mais plutôt une séquence d'attaques, c'est ce que nous appelons un processus d'attaque. Ainsi, lors de la génération des données de test, on construira tout d'abord des scénarios d'attaques, et à chaque étape (séquence) du scénario, on ne prendra qu'un élément de chaque classe.

Pour résumer, les deux problèmes à résoudre sont donc :

- Comment classifier de manière pertinente les attaques ?
- Comment générer des scénarios représentatifs d'attaques ?

Pour répondre à la première question, nous avons commencé par analyser un nombre conséquent de classifications existantes d'attaques [Bishop 1999, Kumar 1995, Hansmann 2003, Lindqvist 1997, Weber 1998, Kendall 1999, Lippmann *et al.* 2000a, Lippmann *et al.* 2000b, Howard 1998, Alessandri04]. Néanmoins, nous avons constaté que la plupart de ces classifications sont centrées sur l'attaquant, c'est-à-dire adoptent le point de vue de l'attaquant (*attacker-centric*, en anglais) [Gad El Rab 2008]. Or ce type d'approches souvent ignore (ou masque) certaines caractéristiques importantes des attaques, telles qu'elles sont vues par l'IDS ou les administrateurs système, alors que ces aspects sont importants dans notre contexte. Notre approche consistait donc à se baser sur les attributs des classifications existantes, en les analysant puis en éliminant ceux qui sont ambigües ou qui ne sont pas pertinents pour l'évaluation des IDS. Les attributs retenus seront accompagnés par une définition claire. Nous avons ainsi abouti à la classification de la figure 26. Notre classification repose sur cinq dimensions (dits aussi attributs). Ces dimensions sont sélectionnées de manière à couvrir les sources, les cibles et les manifestations des attaques, informations qui nous semblent nécessaires et suffisantes pour le test et l'évaluation des IDS.

Nous définissons ces dimensions comme suit :

– *Source* : l'endroit d'où l'attaque a été lancée. Elle possède deux classes : locale et distante.

– *Privilège obtenu* : nous distinguons cinq types de privilèges visés par l'attaquant, les classes "*root*" et "utilisateur" signifient respectivement que l'attaquant a réussi à obtenir l'accès "administrateur" ou "utilisateur" ; la classe "système" permet l'exécution de processus avec les privilèges "systèmes" ; la classe "variable" identifie les attaques qui fournissent l'accès en fonction des privilèges de l'utilisateur de l'application vulnérable exploitée. La classe "aucun" couvre les attaques qui n'ont besoin d'aucun privilège d'accès au système, comme les attaques de reconnaissance (*scans*) ou la plupart des attaques par déni de service (*DoS*).

– *Vulnérabilité* : du point de vue de l'évaluateur, il est intéressant de cibler le système de test le plus pertinent, de bien paramétrer la plateforme de test, mais aussi d'exprimer la relation entre les attaques et les vulnérabilités exploitées ; ceci va en particulier aider à choisir (lors de la phase de test) les attaques qui peuvent exploiter ces vulnérabilités (et qui sont d'ailleurs répertoriées et disponibles dans des bases de données standardisées de vulnérabilités comme CVE ou OSVDB [CVE 2008, OSVDB 2008]), mais également à identifier les failles du système pour une éventuelle correction.

– *Porteur* ou moyen par lequel l'attaque est lancée : il peut s'agir du trafic réseau ou d'actions exécutées directement sur la machine cible et qui n'apparaissent donc pas sur l'interface réseau.

– *Cible* : peut être le système d'exploitation, la mémoire, la pile réseau, le système de fichier ou un processus.

Remarquons que contrairement aux classifications existantes, notre taxonomie tient compte, non seulement des caractéristiques observables de l'attaque (comme c'est le cas des classifications orientées défense), mais aussi des aspects opérationnels, qui sont importants pour l'évaluateur. En effet, la classification que nous proposons fournit les informations essentielles pour la génération des attaques et l'analyse des cas de test. Par exemple, la dimension "source" donne une idée sur l'endroit d'où l'attaque doit être générée pour le test ; de même, la dimension "vulnérabilité" donne une information sur la configuration à avoir lors des tests ; la sévérité des attaques est implicitement décrite à partir de la dimension "privilège" ; etc.

Comme nous l'avons dit précédemment, cette classification caractérise chacune des attaques au niveau élémentaire. Or, dans les cas réels, le test de l'IDS devrait tenir compte des scénarios qui peuvent être composés de plusieurs attaques élémentaires. Néanmoins, force est de constater que l'énumération de tous les scénarios d'attaques s'avère impossible compte tenu du nombre d'attaques éventuelles, des multitudes de variations de chacune des attaques ainsi que des combinaisons possibles de ces attaques et variantes. De plus, lors de l'évaluation des IDS, il faut tenir compte, non seulement des attaques et processus connues, mais aussi des attaques et processus non encore connues.

Il faut donc définir un modèle qui soit le plus possible représentatif du système opérationnel d'un côté, et qui aide à abstraire le comportement des attaquants ainsi que les processus d'attaques d'un autre côté. La question qui se pose maintenant est comment construire un tel modèle alors que dans la pratique, on manque cruellement de données complètes sur les processus d'attaques, comme on manque d'analyses précises du peu de données existantes ? Pour remédier à cela, nous avons analysé les primitives d'exécution d'une quarantaine de maliciels de la liste CME (*Mitre's Common Malware Enumeration list*) [Gad *et al.* 2008] ; celles-ci sont représentatives des attaques les plus dangereuses et

les plus répandues. Nous avons également utilisé d'autres données intéressantes disponibles sur des sites spécialisés comme http://research.eeye.com, http://www.viruslist.com et www.milw0rm.com.

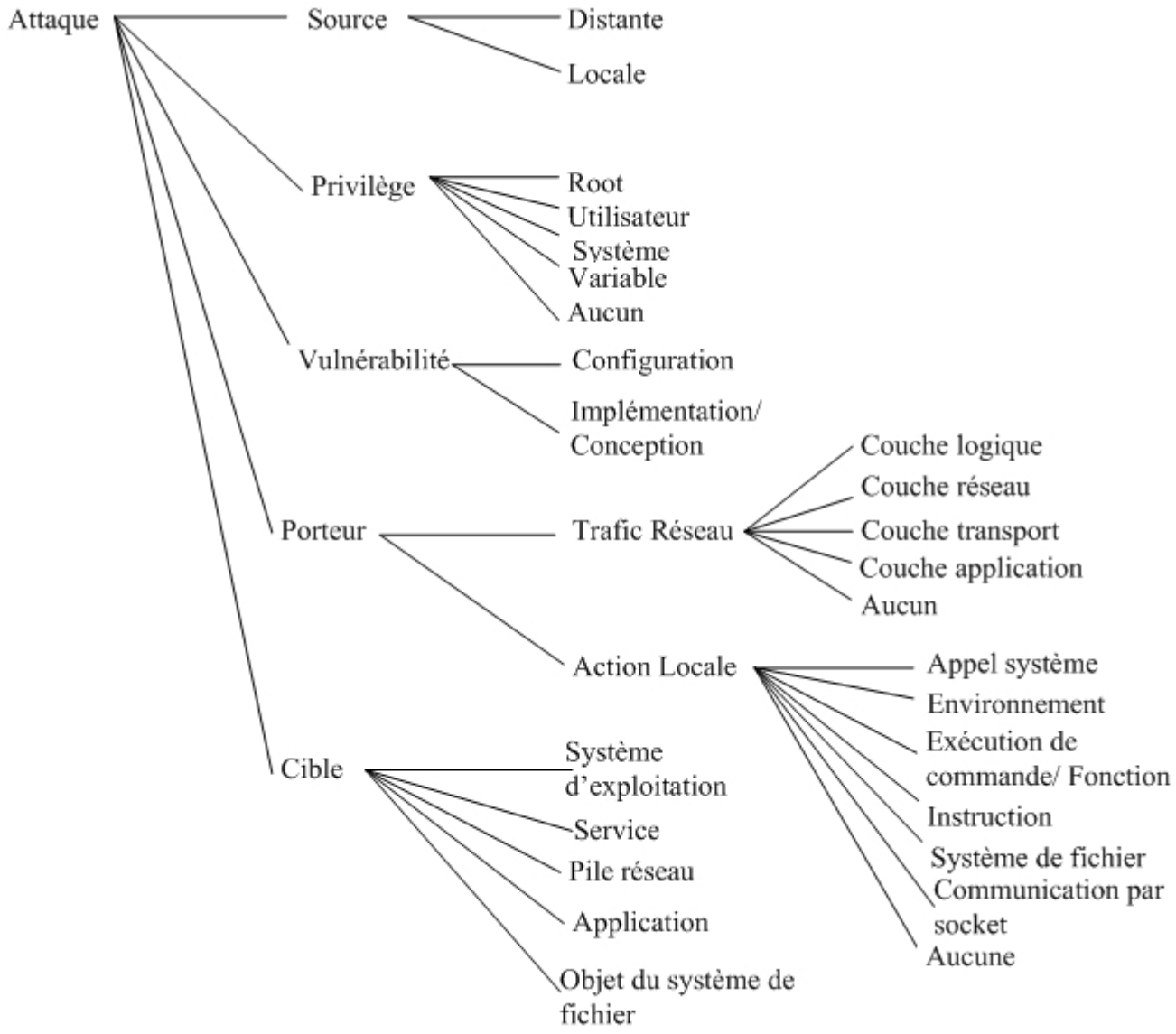


Figure 26 : Nouveau schéma de classification des attaques.

Le premier résultat que nous avons pu constater est que, malgré la diversité de ces maliciels, les étapes suivies peuvent être classées en seulement 8 primitives. Nous avons identifié chaque primitive par un symbole, comme indiqué ci-dessous :

- *R*: Reconnaissance
- *VB*: Fouille des machines ou des réseaux victimes (*Victim Browsing*)

- *EP*: Exécution de programme (*Execute Program*)
- *GA*: Gain d'accès (*Gain Access*)
- *IMC*: Implantation de code malveillant (*Implant Malicious Code*)
- *CDI*: Compromission de l'intégrité (*Compromise Data Integrity*)
- *DoS*: Déni de service (*Denial of Service*)
- *HT*: Effacement des traces (*Hide Traces*)

En regardant de près le fonctionnement de tous ces maliciels, nous avons trouvé que, quelque soit la nature de l'attaque et l'expérience des attaquants, ces derniers suivent généralement le même type de processus (étapes) ; le niveau de l'attaquant se reflète à travers la sophistication et la finesse de l'attaque, la qualité du code, les effets et les dommages causés, etc. Le graphe de la figure 27 schématise le modèle de processus d'attaques résultant de notre analyse. Il permet de générer des scénarios d'attaques à un niveau abstrait. Nous y trouvons des chemins qui correspondent à des scénarios abstraits valides, comme {*R*, *GA*, *VB*, *CDI*, *EP*, *HT*} ou {*R*, *DoS*}. Les chemins non représentés sur le graphe, tel que {R, CDI} ne sont tout simplement pas valides.

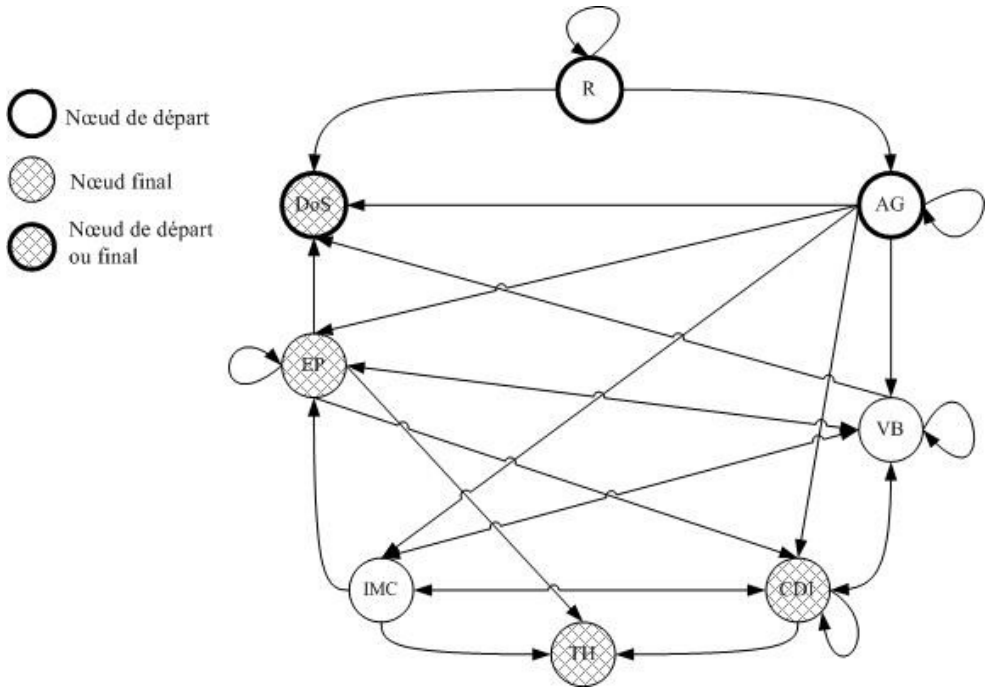


Figure 27 : Modèle de processus d'attaque

Néanmoins, il est clair que cette vue "de haut niveau" n'est pas suffisante pour générer des traces réelles d'attaques. Tout scénario abstrait valide, c'est-à-dire qui correspond à un chemin du graphe de la figure 27, doit en effet être transformé en scénarios exécutables ; ceci devrait passer tout d'abord par une traduction de ces actions primitives (*R*, *GA*, *VB*, CDI, etc.) en des séquences d'actions exécutables. La transformation est faite en associant les primitives avec des commandes concrètes ou des instances

d'exécution d'outils qui réalisent et implémentent ces étapes. Par exemple,

- *ping, traceroute, nessus* et *nmap* sont associés à l'étape ***R***
- *ls*, *ps* et *uname* sont associés à l'étape ***VB***
- *Ssh*, *telnet* et l'exécution d'un exploit *metasploit* sont associés à ***AG***
- *cp*, *rm*, *mv*, éditer un fichier de configuration ou changer les variables d'environnement sont associés à ***CDI***
- *crontab*, *lynx* et *nc* sont associés à ***EP***
- *rm* log, *kill syslog* ou tuer un processus antivrirus sont associés à ***HT***
- *shutdown -halt*, *crash system*, arrêt d'un service (*stop* service) peut être associé avec ***DoS***
- *scp* malveillant, *ftp* malveillant ou l'exécution de metasploit avec une charge utile malveillante peuvent être associés à ***IMC***.

Lors de l'instanciation d'un scénario abstrait (e.g., *R*, *GA*, *VB*, *CDI*, *EP*, *HT*), chaque primitive est remplacée par une ou plusieurs commandes capables de la réaliser. La transformation peut être faite selon différentes techniques : de manière exhaustive, en choisissant aléatoirement une des commandes qui peuvent implémenter chaque étape du scénario, ou en utilisant des algorithmes de transformation plus sophistiqués qui prennent en considération des paramètres comme les résultats des étapes précédentes et le contexte de l'attaque.

Néanmoins, dans l'état, ni les scénarios abstraits, ni leurs instanciations ne permettent de capter les dimensions de notre classification (vulnérabilité exploitée, sévérité de l'attaque, source, privilège requis), alors que ces informations semblent importantes pour l'évaluateur de l'IDS. Pour couvrir ces aspects, nous caractérisons les attaques élémentaires, non seulement selon l'étape du scénario d'attaque (e.g., *R*, *GA*, etc.), mais aussi selon la perspective de classification (e.g., source, privilège, etc.). Ceci permet de sélectionner des attaques élémentaires selon leurs attributs de classification, et de pouvoir exécuter des scénarios exécutables en instanciant les scénarios abstraits selon l'étape de l'attaque. Notons que la caractéristique (attribut) "*étape de l'attaque*" fait le lien entre le modèle de processus d'attaque et la classification des attaques. Par exemple, pour la commande « *ping* » :

- *Source* : distante (même si elle peut être utilisée localement) ;
- Privilège obtenu : aucun ;
- *Vulnérabilité* : configuration ;
- *Porteur* : réseau, niveau transport ;
- *Cible* : Pile réseau ;
- *Étape de l'attaque* : Reconnaissance.

De la même manière, si on prend, toujours à titre d'exemple, l'exploit *ms03_026_dcom* (de *metasploit*) qui exploite une vulnérabilité dans l'interface DCOM lors d'un flux RPC. Celui-ci peut être caractérisé comme suit :

- *Source* : distante ;

- Privilège obtenu : Système ;
- *Vulnérabilité* : conception / implémentation (erreur dans la validation des tailles de données) ;
- *Porteur* : réseau, niveau applicatif ;
- *Cible* : Système d'exploitation ;
- *Étape de l'attaque* : GA (Gain d 'accès);

De cette manière, nous pouvons utiliser conjointement notre classification et notre modèle de processus d'attaques pour instancier les scénarios abstraits, et générer ainsi des scénarios exécutables réels. De plus, toujours dans la perspective d'avoir une évaluation qui soit hautement configurable, on peut éventuellement paramétrer la transformation par un modèle de compétence de l'attaquant. Pour cela, nous pouvons caractériser l'attaquant par son niveau de compétence (*Débutant*, *Intermédiaire*, *Avancé*), son profil (*Hésitant*, *Agressif*), l'ensemble d'outils qu'il possède, et son adresse IP. En outre, on peut également paramétrer les scénarios d'attaques selon des statistiques issues des données collectées à partir de réseaux de pots de miel. Ces statistiques peuvent fournir des informations intéressantes telles que la fréquence d'utilisation de certains outils d'attaques, la fréquence d'occurrence de chaque attaque, les adresses IP les plus utilisées comme source d'attaques, etc., ceci pourrait éventuellement aider à faire les choix d'attaques et de scénarios les plus probants lors de l'instanciation des modèles et la génération de données de test.

Il est important de noter que cette approche itérative de génération des scénarios d'attaques nous a permis également de pallier le problème de l'explosion combinatoire, problème intrinsèque aux approches classiques de génération des scénarios d'attaques. En effet, dans notre modèle, le nombre de cas possibles est fortement réduit, grâce notamment à des contraintes sur les boucles et les relations de précédence (déduites à partir de notre graphe de la Figure 27).

IV. Tests et résultats

Les modèles présentés dans les sections précédentes constituent la base d'un ensemble d'outils que nous avons développés pour l'évaluation des IDS. Notre implémentation consiste en trois parties principales : un *gestionnaire d'évaluation*, un *générateur d'attaques* et un *générateur de trafic de fond*. Alors que ce dernier est basé sur D-ITG, nous avons opté pour une implémentation du gestionnaire d'évaluation et du générateur d'attaques (en utilisant le langage *Ruby*) comme un *plugin* de l'outil de test de pénétration *metasploit.* L'architecture globale de notre outil (*toolkit)* d'évaluation ainsi que les différentes interactions entre ses composants sont illustrées dans la figure 28.

Le gestionnaire d'évaluation contient : une *interface utilisateur* pour personnaliser la configuration, un *planificateur* pour produire les scénarios abstraits, ainsi qu'un *ordonnanceur* pour générer et ordonnancer les scénarios exécutables avant de les distribuer aux agents d'attaques. Le planificateur et l'ordonnanceur utilisent des techniques de programmation logiques par contraintes ; ils ont été développés en utilisant le système de programmation *Mozart*. Celui-ci permet, entre autres, de programmer des stratégies de recherche et de générer des scénarios concurrents et/ou interactifs. La planification des scénarios est transformée en problème de satisfaction de contraintes puis traitée par un *solveur* de contraintes. Une fois le planificateur fournit les scénarios abstraits , l'ordonnanceur assigne les sessions d'attaques aux agents attaquants en tenant compte de paramètres tels que le nombre

d'agents attaquants, le mode d'interférence des sessions (i.e., une ou plusieurs sessions en parallèles), la période d'injection, etc.

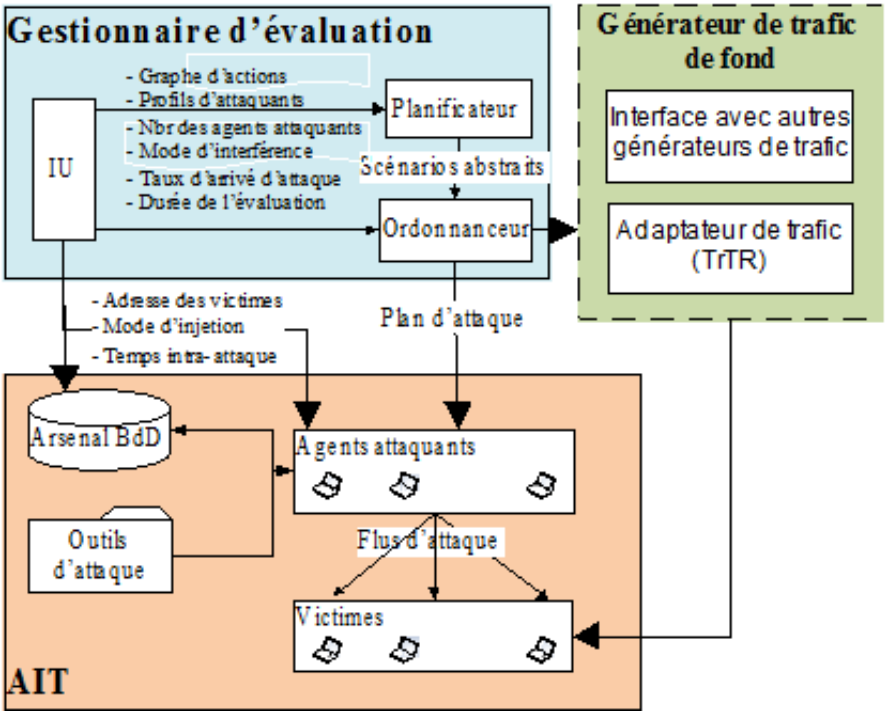


Figure 28 : L'architecture globale de nos outils d'évaluation.

Enfin, l'outil d'injection d'attaques (AIT pour *Attack Injection Tool*) implémente les modèles que nous avons décrit précédemment ainsi que la base de données des attaques élémentaires et des outils d'attaques. L'AIT est flexible et permet à l'évaluateur de sélectionner un ou plusieurs cas particuliers de test et d'ajuster le scénario en précisant certains paramètres comme le niveau de l'attaquant, la plage d'adresse des victimes, le type de vulnérabilités visées (par exemple, ne s'intéresser qu'aux scénarios d'attaques contre une plate-forme sous Windows XP).

Nous avons mis en place plusieurs plate formes de test. La plus basique contient une machine pour l'IDS, une autre pour le générateur de trafic, une ou plusieurs machines attaquantes ainsi que des machines victimes avec différentes configurations (Figure 29).

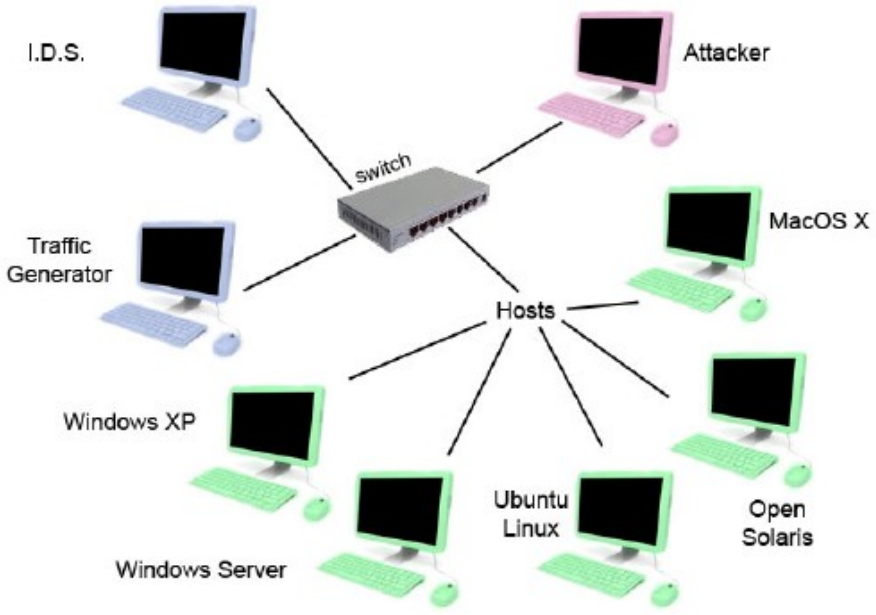


Figure 29 : plate-forme basique de test.

Les résultats de l'évaluation dépendent bien évidement de la plate-forme de test. En effet, nous pouvons varier plusieurs paramètres tels que :

– *l'architecture réseau* : typiquement, nous avons utilisé trois architectures différentes (une simple architecture où les ordinateurs partagent le même réseau ; une architecture avec des VLAN, une DMZ et des passerelles ; une architecture évoluée avec des pare-feux et des sous-réseaux différents) ;

– *le nombre de nœuds sur le réseau et la densité du trafic réseau* : en général, plus on a de nœuds, plus il y aura de données à traiter et à analyser par l'IDS ;

– *le nombre de sessions attaquantes* : il peut s'agir d'un attaquant isolé, de plusieurs attaquants indépendants ou d'attaquants opérant en collaboration (attaques zombies) ; rappelons également que dans notre classification, on tient compte de l'endroit d'où l'attaque est lancée (interne ou externe) ;

– *l'emplacement des sondes de l'IDS* : en effet, les résultats de l'évaluation sont différents selon que la sonde de l'IDS est derrière ou devant le pare-feu par exemple. Des analyses intéressantes peuvent être faites pour voir par exemple quelles sont les attaques qui réussissent à passer le pare-feu mais détectées par l'IDS ; les attaques qui proviennent de d'extérieur ou du réseau interne, les attaques qui visent un VLAN particulier, etc. ;

– *Services et OS déployés*.

Après la mise en place de la plate-forme de test, notre outil permet un paramétrage fin selon notre classification des attaques, par exemple, on l'évaluateur peut paramétrer l'outil de manière à ne lancer que les (i.e., ne s'intéresser qu'aux) attaques distantes, qui exploitent une vulnérabilité spécifique dans une certaine application de Windows XP, qui visent une certaine cible d'un sous-réseau particulier, etc. L'outil sélectionne les attaques (à partir de la base de donnée des exploits de *metasploit*) qui

correspondent aux critères entrées par l'évaluateur (seulement ces attaques et seulement contre les cibles spécifiées), puis les injecte pour voir si l'IDS les détecte ou pas. Un rapport de test est enfin généré automatiquement. Il contient notamment la liste des attaques non détectées ou mal identifiées avec des liens vers les descriptifs de ces attaques au niveau des bases CVE, OSVDB, Bugtraq, Milw0rm et RedHat RHSA.

Nous présentons ici les résultats de nos tests de Snort et Bro avec une architecture classique (celle de la figure 29) avec des machines Windows vista, XP 64Bit Sp1, Xp 32Bit Sp2, Xp 32Bit Sp3 et Windows 2008. Rappelons tout d'abord que puisque Snort est un IDS à base de signatures, il est important de le tester avec différentes bases de règles. Nous avons utilisé les ensembles suivants de règles :

1. *Unregistered rules,* ensemble des règles fournies par les membres non-enregistrés sur le site de Snort,
2. *Registered rules*, ensemble des règles fournies par les membres enregistrés sur le site de Snort,
3. *Community rules*, règles mises à jour régulièrement par la communauté de Snort,
4. *Community + Unregistered rules,*
5. *Community + Registered rules,*
6. *Emerging rules* : règles gratuites fournies par la communauté open-source *Emerging threats* sur le site *www.emergingthreats.net,*
7. *Best* : ensemble de toutes les règles précédentes.

Tableau 12 illustre les résultats des différents tests visant le serveur IIS.

Tableau 12 : Résultats de test pour les attaques visant le serveur IIS.

Exploit	Détection						
	Unregistered	*Registered*	*Community*	*Community + Unregistered*	*Community + Registered*	*Emerging*	*Best*
IIS 5.0 Printer Buffer Overflow	√	√	√	√	√	√	√
IIS 5.0 IDQ Path Overflow	√	√	√	√	√	√	√
IIS 4.0 HTR Buffer Overflow	X	X	X	X	X	X	X
IIS 5.0 WebDAV ntdll.dll Overflow	√	√	√	√	√	√	√
IIS Frontpage fp30reg.dll Chunked Overflow	√	√	√	√	√	√	√
IIS Phone Book Service Overflow	√	X	X	X	√	√	√
IIS nsiislog ISAPI POST Overflow	√	√	√	√	√	√	√
IIS RSA WebAgent Redirect Overflow	X	X	√	√	√	X	√
IIS W3who.dll ISAPI Overflow	X	X	X	X	X	X	X
Taux de détection	**6 sur 9**	**6 sur 9**	**6 sur 9**	**6 sur 9**	**7 sur 9**	**6 sur 9**	**7**

Légende
« √ »Pour détecté et « X » pour non détecté.

Les premiers résultats surprenants peuvent être résumés dans les points suivants :

– malgré le fait que tous les exploits contre IIS sont connus et même assez anciens, aucune des règles gratuites ne permet de les détecter toutes ;

– les différences entre les capacités de détection des différents ensembles de règles restent très minimes ;

– la combinaison de touts les ensembles de règles (nommée *Best* dans nos tests) ne permet finalement de détecter qu'une règle de plus par rapport aux règles fournies par les membres non-enregistrés.

Nous avons ensuite configurer notre outil de test pour ne viser que les attaques distantes visant les vulnérabilités dans les différents services de Windows (DNS, Sql Server, Plug and Play, etc.). Le tableau 13 présente un extrait de nos résultats.

Tableaux 13 : résultats de nos tests visant les services de Windows.

Exploit	Détection						
	Unregistered	*Registered*	*Community*	*Community + Unregistered*	*Community + Registered*	*Emerging*	*Best*
MS IIS 5.0 WebDAV ntdll.dll Overflow	√	√	√	√	√	√	√
Microsoft ASN.1 Libraby Bitstring Heap Overflow	X	√	X	√	√	√	√
Microsoft Server Service NetpwPathCanonicalize Overflow	X	√	X	X	√	√	√
Microsoft Private Communications Transport Overflow	X	X	X	X	X	X	X
Microsoft WINS Service Memory Overflow	X	X	X	X	X	X	X
Microsoft Plug and Play Service Overflow	X	X	X	X	X	X	X
Microsoft Message Queuing Service Path Overflow	X	√	X	X	√	X	√
Microsoft RPC DCOM Interface Overflow	X	X	X	X	X	X	X
Microsoft Server Service Relative Path Stack Corruption	X	X	X	X	X	X	X
Microsoft SQL Server Hello Overflow	X	X	X	X	X	X	X
Microsoft IIS 5.0 IDQ Path Overflow	√	√	√	√	√	√	√

Microsoft IIS Phone Book Service Overflow	√	X	X	X	√	√	√
Microsoft LSASS Service DsRolerUpgradeDownLevelServer Overflow	X	X	√	√	√	√	√
Microsoft RRAS Service RASMAN Registry Overflow	X	X	√	√	√	√	√
Microsoft Service MS06-066 nwapi32.dll	X	√	√	√	√	√	√
Microsoft IIS 4.0 HTR Path Overflow		X	X	X	X	X	X
Microsoft Message Queuing Service DNS Name Path Overflow	X	√	X	X	√	√	√
Microsoft Service MS06-066 nwks.dll	X	√	X	X	√	√	√
Microsoft NetDDE Service Overflow	√	√	√	√	√	√	√
Microsoft SQL Server Resolution Overflow	X	√	√	√	√	√	√
Microsoft IIS 5.0 Printer Host Header Overflow	√	√	√	√	√	√	√
Taux de détection	**5 sur 21**	**11 sur 21**	**8 sur 21**	**9 sur 21**	**14 sur 21**	**13 sur 21**	**14 sur 21**

Globalement, on peut remarquer que :

- les *Registred rules* sont nettement mieux que les *Unregistred rules* et *Community rules ;*
- *Community + registred* ont le même taux de détection que l'ensemble de tous les règles ;
- Même dans les meilleurs cas, 50 % des exploits ne sont pas détectés !

Pour avoir une idée sur les taux de détection pour les exploits les plus récents, nous avons choisi les exploits "*ms08*" de *metasploit*. Aussi surprenant que cela puisse paraître, aucun des ensembles des règles ne permet de détecter ces exploits récents !

Par ailleurs, nous avons testé le seul exploit disponible dans *metasploit* contre le serveur Web "Savant" (http://savant.sourceforge.net/). L'exploit a été détecté, mais mal identifié. En effet, Snort a généré l'alerte "*reverse shell payload*" alors qu'il s'agit de l'exploit "*savant exploit buffer overflow*".

Enfin, nous avons terminé nos tests de Snort par injecter, non pas des attaques élémentaires, mais des scénarios d'attaques. Dans les meilleurs cas, Snort détecte et identifie une partie du scénario (phase de reconnaissance ou d'injection de code maliciel, par exemple) mais n'arrive jamais à identifier le scénario dans sa totalité. Ceci est explicable étant donnée que les IDS à base de signatures, dont Snort, n'ont ni l'architecture, ni la forme de règles pour détecter et identifier un scénario complet d'attaques réalistes.

Nous avons fait les même tests en utilisant Bro, un IDS réseau open-source. Le premier type de tests contre IIS a donné les résultats du tableau 17. On remarque que Bro détecte 5 sur les 9 exploits injectés,

ce qui signifie qu'il est meilleur que Snort (quelque soit la base de signature utilisée dans Snort). Par contre, en lançant les tests contre les services Windows, Bro ne détecte aucune des attaques !

Tableau 19 : résultats des tests de Bro pour la détection des attaques contre IIS.

Vulnérabilité	Détection
IIS 5.0 Printer Buffer Overflow	X
Microsoft IIS 5.0 IDQ Path Overflow	√
IIS 4.0 HTR Buffer Overflow	X
IIS WebDAV ntdll.dll Overflow	X
IIS Frontpage fp30reg.dll Chunked Overflow	√
Microsoft IIS Phone Book Service Overflow	√
IIS nsiislog.dll ISAPI POST Overflow	√
IIS RSA WebAgent Redirect Overflow	√
IIS w3who.dll ISAPI Overflow	X

V. Conclusions

Si aujourd'hui, les produits de détection d'intrusion sont légions, les techniques d'évaluation des IDS sont encore dans un état embryonnaire. Pourtant, l'étude de ce problème fondamental devrait nous aider à identifier les points les plus critiques à surveiller ou à corriger dans les IDS, à pouvoir comparer les fonctionnalités, les performances ainsi que les caractéristiques de plusieurs IDS, et à estimer s'il est rentable de mettre en œuvre telle ou telle défense supplémentaire.

Très peu de recherches se sont intéressées à ce thème. Quelques tentatives de grande envergure ont bien été entreprises mais elles possèdent plusieurs faiblesses, notamment l'*utilisation d'approches non systématiques,* la *non-représentativité* des données de test et la *non-pertinence des métriques.*

Afin de pallier l'ensemble de ces limitations, nous avons commencé, essentiellement dans la thèse de M. Gad El Rab que je co-encadre, par proposer une méthodologie systématique d'évaluation des IDS [Gad & Abou El Kalam 2006]. Ensuite, nous avons proposé une classification des attaques ainsi qu'un modèle de scénario d'attaques [Gad *et al.* 2008a, Gad *et al.* 2008b]. La combinaison de ces deux modèles nous permet en effet de générer, de manière réfléchie, des scénarios concrets d'attaques à partir de scénarios abstraits reflétant les besoins de l'évaluation.

Utilisant cette base théorique, nous avons ensuite développé un ensemble d'outils pour l'évaluation des IDS : un *gestionnaire d'évaluation*, un *générateur d'attaques* et un *générateur de trafic de fond* basé sur D-ITG [Gad *et al.* 2009].

Ensuite, nous avons utilisé nos outils pour planifier, générer et injecter des attaques dans des plateformes intégrant Snort dans un premier temps, puis Bro dans un deuxième temps. Cette étude comparative nous a permis d'identifier quelques-unes des forces et faiblesses de chacun des deux IDS.

Notons que nos modèles et outils sont assez génériques et hautement paramétrables, permettant

ainsi de tester n'importe quel IDS, voir même d'autres outils de sécurité comme les pare-feux.

Enfin, signalons que les travaux décrits dans ce chapitre seront étendus dans la thèse de K. Salih que je co-encadre et qui a démarré en septembre 2009 ; celle-ci s'inscrit dans le cadre du réseau d'excellence européen NEWCOM++ (*European Network of Excellence in Wireless Communications*) et s'intéresse à l'évaluation des IDS en environnement sans fil.

Conclusions et perspectives

Les projets de recherche auxquels nous avons participé ces cinq dernières années sont centrés sur les infrastructures et réseaux critiques, notamment le réseau d'électricité européen et les réseaux avioniques embarqués. Jadis, ces réseaux étaient plutôt fermés, mais pour des raisons fonctionnelles et économiques évidentes, leurs systèmes d'information commencent à s'ouvrir à des architectures, protocoles et applications plus ou moins vulnérables. Or, dans de tels infrastructures et réseaux, une simple défaillance (due à une faute accidentelle ou à une action malveillante) peut causer des dégâts humains et financiers catastrophiques. Ainsi, non seulement il est question de sécuriser ces applications (e.g., en les protégeant notamment contre les attaques potentielles), mais il faut également être capable de justifier notre confiance dans les mécanismes de sécurité déployés, c'est à dire pouvoir valider, évaluer et vérifier la sécurité.

Afin d'assurer la sécurité, la définition d'une politique rigoureuse de sécurité s'impose. Celle-ci doit identifier de manière claire et non-ambigüe les objectifs de sécurité à assurer ainsi que les règles de sécurité qui régissent la manière dont les ressources sont utilisées protégées dans le système. Ensuite, d'une part il faut vérifier la cohérence de cette politique pour détecter et résoudre les conflits éventuels, et d'autres part, il faut la mettre en œuvre par des mécanismes de sécurité appropriés tels que les VPN et règles de pare-feux mais aussi d'en surveiller les violations, par exemple avec les outils de détection d'intrusions. Toutefois, de tels outils de sécurité peuvent eux même délivrer un service non conforme au service attendu. Il faut donc être capable de les évaluer afin de corriger leurs éventuels vulnérabilités ou de décider s'il faut déployer telle ou telle défense supplémentaire.

S'intéressant à ces différents maillons de sécurité, nos travaux de recherche traitent des points suivants :

– Définition de PolyOrBAC, un nouveau cadriciel de sécurité pour les systèmes dynamiques, collaboratifs et multi-organisationnels. PolyOrBAC possède la particularité de pouvoir gérer de manière sécurisée la collaboration de plusieurs organisations mutuellement suspicieuses tout en sauvegardant une certaine autonomie et indépendance de chacune des organisations. PolyOrBAC utilise : (1) OrBAC pour spécifier localement les politiques de sécurité, (2) les mécanismes de service web pour les communications en environnement collaboratif, (3) la notion d'image de service web (*resp*. d'utilisateur virtuel) pour représenter (virtualiser) au niveau de l'organisation cliente (*resp*. prestataire) les services Web (*resp*. utilisateurs) distants, (4) les mécanismes de vérification des interactions de service Web grâce à des politiques-contrats définies à l'aide d'automates temporisés qui garantissaient le respect des contrats de collaboration établis par les organisations concernées.

– Enrichissement des politiques de contrôle d'accès à travers une modélisation du concept de recommandation et une définition de la syntaxe, la sémantique, les conditions de vérité, les règles d'inférences ainsi que l'axiomatique qui sont liés à ce concept.

– Utilisation de la programmation logique par contraintes pour la vérification de la cohérence d'une politique de sécurité, en particulier pour la résolution et la détection de conflits de modalités

– Proposition du protocole Q-ESP, une amélioration d'IPSec qui assure à la fois des besoins de sécurité et de QoS. Ce protocole est implémenté au niveau du noyau NetBSD puis évalué vis-à-vis de la sécurité et des métriques classiques de QoS.

– Définition d'une méthodologie d'évaluation des outils de sécurité ainsi qu'une classification des attaques et une proposition d'un modèle de processus d'attaques ; ces techniques ont été implémentées et utilisées pour la comparaison (*benchmarking*) de systèmes de détection d'intrusions.

L'ensemble de ces travaux ouvre d'énormes voies de recherche à moyen et à long terme et dont certaines sont déjà entamées dans le cadre de plusieurs projets européens.

Extensions possibles de PolyOrBAC

Étendre l'application de PolyOrBAC à d'autres infrastructures et applications collaboratifs :

En effet, les architectures des réseaux d'électricité européens se dirigent plutôt vers l'intégration de nouvelles entités dissimulées, mobiles et ayant certains rôles spécifiques, par exemple, la distribution de l'énergie à travers des voitures électriques à panneaux solaires ou à travers des réseaux électriques distribués et intelligents connus sous le nom de "*smart grid*" en anglais. Plus généralement, il serait intéressant de voir si PolyOrBAC s'applique bien à d'autres infrastructures de collaboration telles que l'infrastructure économique ou bancaire nationale. Il serait néanmoins probablement nécessaire de prévoir de nouveaux mécanismes, notamment de tolérance aux fautes, en raison de certaines spécificités de ces réseaux et infrastructures.

Prise en compte de la notion d'intégrité :

Dans des applications telles que les infrastructures critiques, l'intégrité s'avère plus importante que la disponibilité. Dans le réseau d'électricité par exemple, les centres de distributions ne cherchent pas particulièrement à cacher l'existence du délestage ; par contre, ils veulent être certains qu'il s'est effectué correctement quand la fonction correspondante est invoquée. Des exemples similaires peuvent être donnés dans les domaines avioniques et bancaires. Partant de ce besoin fort d'intégrité, notre approche peut être étendue pour contrôler les flux d'informations entre des processus ou organisations de criticités différentes.

Les politiques classiques d'intégrité (Biba et ses dérivées) essayent de résoudre ce type de problèmes en interdisant le passage de flux de tâches ayant un niveau de criticité inférieure vers des tâches de plus haut niveau de criticité. Or, dans certains cas réels, on a besoins que les communications se fassent également dans l'autre sens (depuis des tâches moins critiques vers des tâches plus critiques). Dans ce cas, il serait plus intéressant d'étendre OrBAC (et par conséquent PolyOrBAC) en se basant plutôt sur des modèles plus récents d'intégrité comme celui de Totel *et al.* Ce dernier permet notamment des flux depuis des processus moins critiques vers des processus plus critiques, si ces flux sont validés par des moyens adéquats grâce à des mécanismes pour la tolérance aux fautes par exemple [Totel *et al.*, 1998].

Par ailleurs, la notion d'intégrité peut porter non seulement sur les criticités des tâches et processus, mais peut être étendue aux crédibilités des organisations, par exemple, comment gérer les flux entre des organisations de crédibilités différentes. Ces niveaux peuvent éventuellement varier selon l'historique des actions des organisations ; e.g., si une certaine organisation ne satisfait pas ses obligations ou si ses

utilisateurs abusent de leurs droits, on peut imaginer que son niveau de crédibilité baisse. De cette manière, la vérification des règles se fait, non seulement selon les règles OrBAC, mais aussi selon les règles gérant le contrôle des flux inter-organisationnels, ceux-ci tiennent compte des niveaux de crédibilité.

Au niveau OrBAC, nous pensons que l'intégrité peut être traduite par des règles d'obligation et d'interdiction. En effet, une des facettes de l'intégrité est d'empêcher une modification par des utilisateurs non autorisés ; ceci peut éventuellement être spécifié à travers une règle d'interdiction. De la même manière, faire en sorte qu'aucun utilisateur ne puisse empêcher la modification légitime de l'information peut correspondre à une règle d'interdiction. Dans le même sens, empêcher une modification incorrecte par des utilisateurs autorisés, peut être vue comme l'obligation de fournir un service et que ce service doit être correct. Une vérification (que le service est fourni et qu'il est correct) s'impose donc ; ce qui met en évidence le lien entre l'intégrité et l'obligation.

***Prise en compte de la notion de disponibilité* :**

La disponibilité doit être considérée non seulement du point de vue des fautes accidentelles, mais aussi en prenant en compte les actions malveillantes. Ceci peut être traité par l'utilisation entre autres de règles d'obligation, afin de garantir que l'organisation prestataire fournira les mécanismes adéquats de sécurité et de tolérance aux fautes (e.g., fournir suffisamment de ressources et interdire leurs monopolisations, détecter les défaillances des processus et des canaux) pour répondre aux besoins et demandes des organisations clientes, même en cas d'événements imprévus tels que des défaillances ou les attaques sur le système.

Des investigations doivent ainsi être menées non seulement sur la façon de spécifier formellement le concept de disponibilité et le problème du déni de service au moyen d'une politique de disponibilité, mais aussi sur les mécanismes capables d'implémenter cette propriété.

Harmonisation des concepts de PolyOrBAC avec ceux connus dans le domaine des applications distribués :

En particulier, les notions d'image de services Web et d'utilisateurs virtuels de PolyOrBAC sont très similaires aux *proxies* de la norme CORBA (*Common Object Request Broker Architecture*) et aux *stub* (en français par *souche*) et *skeleton* (*squelette*) de la technologie Java RMI (*Remote Method Invocation*). De cette manière PolyOrBAC pourrait être tout simplement combiner OrBAC avec les technologies des services Web et des systèmes distribués, sans forcément utiliser les nouvelles notions d'image de service Web et d'utilisateur virtuel. Ceci peut éventuellement aider à une large acceptation, diffusion et déploiement de PolyOrBAC, voir même une normalisation de ses concepts.

Extensions possibles de nos travaux sur les recommandations

Extension d'OrBAC afin de gérer les recommandations :

De la même manière qu'OrBAC définit des règles de permissions, obligations et interdiction, il peut également utiliser des règles de recommandations. En utilisant l'axiomatique et les règles d'inférences de notre langage de spécification de recommandations, il serait possible de raisonner sur les différents types de règles OrBAC, détecter et résoudre les conflits éventuels, et pouvoir interroger la politique OrBAC, même en présence de recommandations. Enfin, il serait intéressant d'intégrer la notion de recommandation dans les outils existants comme le moteur JoRBAC, ou de manière plus globale, dans

des langages comme Prolog.

Gestion des recommandations dans les environnements distribués :

La notion de recommandation, telle que nous l'avons décrit dans notre travail, ne concerne qu'une seule autorité. Une extension possible serait de considérer un environnement avec plusieurs autorités (ou agents), chacune avec ses propres axiomes et règles, et de dériver ensuite des axiomes, formules et règles d'inférences pour tout le système. Par exemple, soit *n* autorités notées {1, … *n*}, $\mathbf{O}_i\phi$ est une formule qui signifie que ϕ est obligatoire selon l'autorité *i*. On pourrait essayer de prouver la vérité (ou pas) de formules du type $\mathbf{O}_1\phi_1 \wedge ... \wedge \mathbf{O}_n\phi_n \rightarrow \mathbf{P}_j(\phi_1 \wedge ... \wedge \phi_n)$; celle-ci signifie que si l'autorité 1 stipule que ϕ_1 est obligatoire, et…, et si l'autorité *n* stipule que ϕ_n est obligatoire , alors (ϕ_1 et … et ϕ_n) est permis pour n'importe quelle autorité appartenant à {1, … *n*}. Et si cette formule semble a priori acceptable, d'autres comme $\mathbf{O}_1\phi_1 \wedge ... \wedge \mathbf{O}_n\phi_n \rightarrow \mathbf{R}_j(\phi_1 \wedge ... \wedge \phi_n)$ ou $\mathbf{R}_1\phi_1 \wedge ... \wedge \mathbf{R}_n\phi_n \rightarrow \mathbf{P}_j(\phi_1 \wedge ... \wedge \phi_n)$ semblent moins intuitives et nécessite donc des preuves.

Amélioration de la notion de recommandation :

Après plusieurs réflexions autour de ce concept de recommandation, nous pensons qu'il peut en fait revêtir plusieurs acceptions. Outre les deux visions que nous avons publiées autour de cette notion (dont une est présentée dans ce rapport), on pense qu'une recommandation peut également être vue comme une sorte d'obligation avec des exceptions spécifiques. Plus précisément, "A est recommandé" peut être interprétée par "A est obligatoire" ou "Si A n'est pas réalisé, il faut satisfaire d'autres obligations: B, C, …".

Extensions possibles de nos travaux sur Q-ESP :

Finalisation et normalisation

Nous orientons à présent nos travaux dans ce domaine sur le développement de l'implémentation du protocole Q-ESP au niveau du noyau NetBSD pour la version 6 d'IP (IPv6), et la mise à jour des outils IKE de configuration dynamique des SA (au niveau de *Racoon*). Nous nous penchons également à la mise au point de patchs pour *ipfilter* [40], *ipnat* [41] et plus généralement sur l'ensemble des mécanismes de QoS basés sur les *MF Classifier*. En effet, le développement du support d'IPsec Q-ESP par *ipfilter* et *ipnat* est incontournable en vue d'une éventuelle intégration dans les systèmes BSD présents et futurs. Enfin, un draft IETF (*Internet Engineering Task Force*) est rédigé pour Q-ESP et des contacts sont entrepris avec CISCO et Sun Microsystems pour une éventuelle normalisation. Plus généralement, nous souhaitons contribuer au développement d'une nouvelle génération de protocoles destinés à améliorer le comportement du réseau vis-à-vis des flux applicatifs et de garantir à la fois la sécurité et la qualité de service.

Adaptation du protocole Q-ESP à d'autres types de réseaux et applications :

Assurer à la fois la sécurité et la QoS est une préoccupation majeure de la plupart des applications émergentes telles que MPLS, VoIP, NGN, réseaux satellitaires et réseaux avioniques. Il serait donc intéressant d'adapter Q-ESP à chacune de ses applications. Par exemple, le problème de découverte, récupération et exploitation des boîtes noires des avions, problème récurent et d'actualité, peut être résolu par l'envoie en temps réel d'informations sur le vol par voie satellite. Toutefois, ceci est

confronté aux problèmes évidents de sécurité et QoS. Une adaptation de Q-ESP peut tout à fait répondre à ce type de besoins.

Extensions possibles de nos travaux sur l'évaluation des outils de sécurité

Raffinement de la classification et du modèle de processus d'attaques :

Il est clair que nos travaux sur l'évaluation des IDS restent dans un état embryonnaire (au même titre que tous ceux qui existent dans ce domaine) . Tout d'abord, un des problèmes de la caractérisation des attaques et la distinction entre les données malicieuses et les données bénignes non-malveillantes, à la fois dans le trafic réseau et les données des hôtes. Ceci ne peut se faire de manière isolée (commande par commande), étant donné que certaines commandes servent aussi bien à faire des actions légitimes que des actions malveillantes, mais dans le cadre de processus d'attaques et d'enchaînement d'actions. Des recherches dans ce sens nous aideront, non seulement à recueillir des informations importantes sur les données malveillantes et les données bénignes, mais aussi à améliorer les techniques de détection.

Par ailleurs, la classification que nous proposons doit être revue régulièrement pour prendre en compte plus d'attaques, notamment celles nouvellement apparues. Le but ultime étant d'arriver à une version complète afin de pousser son intégration à des outils existants tels que *metasploit* ou à des standards et bases de données internationales tels que CME, CVE et OSVDB. Des contacts sont d'ailleurs déjà entrepris avec l'équipe de développement et de maintenance de *metasploit*. Une fois la classification riche et stabilisée, il conviendrait de définir des profils d'attaques qui soient les plus représentatifs et qui peuvent servir à tester des outils de sécurité.

Dans le même sens, étant donné que les outils, techniques et processus d'attaques évoluent vite, notre modèle de processus d'attaque doit être mis à jour, validé et raffiné régulièrement afin de couvrir l'ensemble des scénarios possibles, probables et envisageables. En particulier, il faut tenir compte des processus d'attaques multi-sauts qui impliquent plusieurs attaquants opérant en groupe, à différents moments, depuis plusieurs machines et visant plusieurs cibles accessibles à partir de la première victime attaquée.

Enfin, afin d'améliorer le modèle de compétence de l'attaquant, nous sommes en train de mettre en place un pot de miel haute interaction. Un déploiement intelligent et durable pourrait nous aider à extraire des données importantes pour enrichir le modèle statistique. Toujours dans le même but d'enrichissement de notre modèle, il serait intéressant de puiser dans des techniques issues d'autres domaines comme l'intelligence artificielle, l'ergonomie, voir même la psychologie et la sociologie.

Adaptation de notre travail aux environnements sans fil :

Dans un environnement sans fil, on peut distinguer deux types d'IDS réseau : les IDS classiques (initialement conçus pour l'environnement filaire) qui ont été actualisés avec une base de signature spécifique afin de détecter les attaques contre les réseaux sans-fil, ce type d'IDS est dit « *Wired IDS* »; et les IDS qui ne fonctionnent que dans un environnement sans fil, ce type d'IDS est appelé « *Wireless IDS* ». Néanmoins, en plus des limites qu'on trouve chez les IDS classiques, les deux types d'IDS présentent actuellement d'autres faiblesses, essentiellement dues à leur jeunesse et au manque de travaux traitant de leurs évaluations. Il devient donc nécessaire d'évaluer les deux types d'IDS en environnement sans-fil, afin de voir leurs faiblesses, forces, robustesses, performances, mais aussi afin de pouvoir les comparer (*benchmarking*).

Ceci nécessite une évaluation basée à la fois sur l'analyse du modèle de l'IDS et sur le test de l'IDS (dans le sens logiciel) tout en tenant compte des spécificités des attaques et des architectures des réseaux sans-fil. Il est ainsi clair qu'avant de passer à l'expérimentation, un travail conséquent de recherche doit être mené, notamment pour adapter nos différents modèles (processus d 'attaques, classification des attaques, etc.) aux environnements sans-fil.

Références bibliographiques

[Abou El Kalam 2008a] A. Abou El Kalam, “A Research Challenge in Modeling Access Control Policies: Modeling Recommendations”, *IEEE International Conference on Research Challenges in Information Science*, 3-6 juin 2008, Marrakech, Maroc.

[Abou El Kalam 2008b] A. Abou El Kalam, "Specification & Enforcment of Access Control in Information & Communication Systems" 3rd *IEEE International Conference on Information & Communication Technologies: From Theory to Applications* (IEEE ICTTA), 7 – 11 avril, Damas, Syrie, 2008.

[Abou El Kalam 2008c] Anas Abou EL Kalam, "Etude de la sécurité dans le réseau avionique ADCN", WP.2.1, Rapport interne de contrat, IRIT, Toulouse, 2008.

[Abou El Kalam 2008d] Anas Abou EL Kalam, "Sécurité des réseaux avionique ADCN : étude des vulnérabilité", WP.2.1, Rapport interne de contrat, IRIT, Toulouse, 2008.

[A. Abou El Kalam et al. 2008e] A. Abou El Kalam, A. Baina, H. Beitollahi, A. Bessani, A. Bondavalli, M. Correia, A. Daidone, G. Deconinck, Y. Deswarte, F. Grandoni, H. Moniz, N. Neves, T. Rigole, P. Sousa, P. Verssìmo, "specification of services and protocols for the Crutial Architecture", 2008, 118 pp.

[Abou El Kalam 2009] Anas Abou EL Kalam, "Sécurité des réseaux avionique ADCN : étude des solutions", WP.2.1, Rapport interne de contrat, IRIT, Toulouse, 2008.

[Abou El Kalam *et al.*, 2003] Anas Abou El Kalam, Rania El Baida, Philippe Balbiani, Salem Benferhat, Frédéric Cuppens, Yves Deswarte, Alexandre Miège, Claire Saurel, Gilles Trouessin, “Organization Based Access Control”, *4th IEEE International Workshop on Policies for Distributed Systems and Networks* (Policy'03), Côme, Italie, juin 2003, pp. 120-131. <http://www.orbac.org/publi/OrBAC/OrBac.pdf>

[Abou El Kalam *et al.*, 2007a] Anas Abou El Kalam, Yves Deswarte, Amine Baina, Mohamed Kaaniche, "Access Control for Collaborative Systems: A Web Services Based Approach”, *IEEE International Conference on Web Services* (ICWS 2007), 2007, pp. 1064-1071. <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4279707>

[Abou El Kalam *et al.*, 2007b] Anas Abou El Kalam, Amine Baina, Hakem Beitollahi, Alysson Bessani, Andrea Bondavalli, Miguel Correia, Alessandro Daidone, Geert Deconincc, Yves Deswarte, Fabrizio Grandoni, Nuno Neves, Tom Rigole, Paulo Sousa, Paulo Veríssimo, «Preliminary Architecture Specification», Deliverable D4, The CRUTIAL Project, FCUL, Lisbone, Portugal, 2007, 106 pp. <http://crutial.cesiricerca.it/content/files/Documents/Deliverables%20P1/WP4-D4-final.pdf>

[Abou El Kalam *et al.*, 2008a] Anas Abou El Kalam, Jérôme Ermont , Yves Deswarte, Specification

and Verification of Security Properties of e-Contracts, Rapport LAAS n°08311, juin 2008, 7p. <http://www.laas.fr/~deswarte/Publications/08311.pdf>.

[Abou El Kalam *et al.*, 2009a] Anas Abou El Kalam, Yves Deswarte, Amine Baïna, Mohamed Kaâniche, PolyOrBAC: a Security Framework for Critical Infrastructures, Rapport LAAS N°09087, 28 pp., *International Journal on Critical Infrastructure Protection*, Elsevier, vol. 2(4), Décembre 2009, 37pp, LNCS.

[Abou El Kalam *et al.*, 2009b] A. Abou El Kalam, M. Mostafa, C. Fraboul, "A New Protocol for Security and QoS in IP Networks", *International Journal of Information and Computer Security* (*IJICS*), InderScience Publishers, 18pp, décembre 2009.

[Abou El Kalam & Balbiani 2009] A. Abou El Kalam, P. Balbiani, "*A Policy Language for Modelling Recommendations*", *IFIP TC*-11 *International Information Security Conference*, *(IFIP SEC* 2009), Cyprus, 18-20 juin 2009, Springer.

[Abou El Kalam & Deswarte, 2006a] Anas Abou El Kalam, Yves Deswarte, "Multi-OrBAC: a New Access Control Model for Distributed, Heterogeneous and Collaborative Systems", *8th IEEE International Symposium on Systems and Information Security* (SSI 2006), Sao Paulo, Brésil, 8-10 novembre 2006. < http://hal.archives-ouvertes.fr/ccsd-00086523>

[Abou El Kalam & Deswarte, 2009a] Anas Abou El Kalam, Yves Deswarte, "Critical Infrastructures Security Modeling, Enforcement and Runtime Checking", *3rd InternationalWorkshop on Critical Information Infrastructures Security* (CRITIS'08), 13-15 octobre, 2008, Frascati, Italie. Lecture Notes in Computer Science, LNCS 5508, 2009.

[Abou El Kalam & Deswarte, 2009b] A. Abou El Kalam, Y. Deswarte, "*Poly-OrBAC: An Access Control Model fior Inter-Organizational Web Services*", Handbook of Research on Semantic Technologies and Web Services, ISBN: 978-1-60566-650-1, May 2009, IGI-Global Editor, <http://www.igi-global.com/reference/details.asp?ID=34405>

[Abou El Kalam & Gad 2006a] A. Abou El Kalam, M. Gad, "Evaluation des systèmes de détection d'intrusion", *Revue de la REE*, *Numéro spécial* « *Risques et Sécurité des Réseaux et des systèmes à composante logicielle* », N° 6/7, pp. 24-32, juin-juillet 2006.

[Abou El Kalam & Gad 2006b] A. Abou El Kalam, M. Gad, "Une nouvelle méthodologie pour l'évaluation des systèmes de détection d'intrusion", *5th Conference on Security and Network Architectures (SAR'06)*, Seignosse, 6-9 Juin 2006.

[Abou El Kalam & Ouabiba 2005] A. Abou El Kalam, M. Ouabiba, "Une nouvelle approche pour la détection et la résolution de conflits de sécurité dans les systèmes multi-organisationnels", *Premières Journées Francophones de Programmation par Contraintes (JFPC'05)*, Lens, 8-10 juin 2005.

[Alessandri04] D. Alessandri, "Attack-Class-Based Analysis of Intrusion Detection Systems", *Ph.D. Thesis*, University of Newcastle upon Tyne, School of Computing Science, Newcastle upon Tyne, UK, 2004.

[Alur & Dill 1994] Rajeev Alur, David L. Dill, "A Theory of Timed Automata", *Theoretical Computer Science*, Vol. 126, No. 2, 1994, pp. 183-235. <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.51.1093>

[Aqvist 1966] Aqvist, "Next and Ought, alternative foundations for Von Wright's tense-logic, with an

application to deontic logic". *Logique & Analyse*, vol. **9** (1966) 231–251.

[Baïna *et al.*, 2008a] Amine Baïna, Anas Abou El Kalam, Yves Deswarte, Mohamed Kaâniche, "A Collaborative Access Control Framework for Critical Infrastructures", *Critical Infrastructure Protection* II, Ed. Mauricio Papa & Sujeet Shenoi, *2nd Annual IFIP 11.10 Conference on Critical Infrastructure Protection*, 16-19 mars 2008, Arlington, États-Unis, Springer, IFIP Series, ISBN 978-0-387-88522-3, pp.189-204, disponible à <http://www.springerlink.com/content/8l863582312285n1/fulltext.pdf>

[Baina *et al.* 2008b] A. Baïna, A. Abou El Kalam, Y. Deswarte, M. Kaaniche, "Access Control for Cooperative Systems: A Comparative Analysis", *International Conference on Risks and Security of Internet and Systems*, Tozeur - Tunisia, 28-31 October 2008, IEEE.

[Benferhat *et al.* 2003] S.Benferhat, R.El Baida, and F.Cuppens, "A Stratification-Based Approach for Handling Conflicts in Access Control", 8th *ACM Symposium on Access Control Models and Technologies* (SACMAT'03), Lake Come, Italie, juin 2003.

[Betini *et al.* 2002] C. Bettini, S. Jajodia, X. S. Wang et D. Wijesekera, "Obligation Monitoring in Policy Management", *International Workshop, Policies for Distributed Systems and Networks* (*Policy*), Monterey, California, 5-7 June 2002, IEEE Computer Society Press, pp. 2-12.

[Bertino *et al.*1996] E. Bertino, S. Jajodia et P. Samarati, "Supporting Multiple Access Control Policies in Database Systems" *IEEE Symp. on Security and Privacy*, Oakland, 1996.

[Bishop 1999] M. Bishop, "Vulnerabilities Analysis", *Proceedings of the 2nd International Workshop on Recent Advances in Intrusion Detection (RAID'99)*, West Lafayette, Indiana, USA, 1999.

[Blake *et al.*1998] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, "An Architecture for Differentiated Services". RFC 2475, décembre 1998.

[Bieber & F. Cuppens 1991] P. Bieber, F. Cuppens, "A definition of secure dependencies using the logic of security", *Computer Security Foundations Workshop*. IEEE (1991).

[Borg *et al.* 1999] N. Borg,E. Savanberg, O. Schelén, "Efficient Multi-Field Packet Classification for QoS Purposes"; 1999; *Seventh International Workshop on Quality of Service* (IWQoS); juin 1999, London, pp. 109-118, ISBN: 0-7803-5671-3.

[Braden *et al.* 1994] Braden, R., D. Clark, and S. Shenker, "Integrated Services in the Internet Architecture: an Overview". RFC 1633, juin 1994.

[Glasgow *et al.* 1992] J. Glasgow, G. MacEwan, and P. Panagaden. "A logic for reasoning about security", *ACM Transactions on Computer Science*, 10, 226264 (1992).

[Catach 1989] L. Catach, *Les logiques multimodales*, Thèse de doctorat, Université de Pierre et Marie Curie (Paris 6), Paris, France, 1989, 312 pp.

[Chellas 1980] B.F. Chellas, "*Modal Logic : An Introduction*", Cambridge University Press, 1980, ISBN 0-521-29515-7, 295 pp.

[Cholvy & Demolombe 1986] L. Cholvy and R. Demolombe, "Querying a rule base", *first International Conference on Expert Database Systems*, Charleston, 1986.

[Colmerauer 1990] A. Colmerauer, "An introduction to Prolog III", *Communication of the ACM*, vol33, N°7, pp. 70-90, juillet 1990.

[Common Criteria 2006] Common Criteria for Information Technology Security Evaluation, Version 3.1 Revision 1, *Part* 1*: Introduction and general model*, CCMB-2006-09-001, 86 p., Septembre 2006.

[CORBA 98] OMG, “The Common Object Request Broker : Architecture and Specification, revision 2.2”, USA, Février 1998, OMG TC Document formal/98-07-01 <http://www.omg.org/corba/corbiiop.htm>

[Cuppens-Boulahia 2007] Nora Cuppens-Boulahia, “*Expression, Déploiement, Analyse et Administration de Politiques de Sécurité*”, Rapport d'Habilitation à Diriger les Recherches, École Nationale Supérieure des Télécommunications, de Bretagne, Septembre 2007.

[Cuppens *et al.* 2004] Frédéric Cuppens, Nora Cuppens-Boulahia, Thierry Sans, Alexandre Miège, "A formal approach to specify and deploy a network security policy", 2nd W*orkshop on Formal Aspects in Security and Trust* (FAST), Toulouse, France, Août 2004.

[Cuppens *et al.* 2005] Frédéric Cuppens, Nora Cuppens-Boulahia, Thierry Sans, "Nomad : A Security Model with Non Atomic Actions and Deadlines", *The computer security foundations workshop* (CSFW), Aix en Provence, France, juin *2005* . IEEE Computer Society Press.

[Cuppens & Cuppens-Boulahia 2008] F. Cuppens & N. Cuppens-Boulahia, “Modeling Contextual Security Policies”, *International Journal of Information Security* (IJIS), vol 7, N° 4, Août 2008, Springer, 33 pp.

[CVE 2008] Mitre's Common Vulnerability and Exposure (CVE) (2008): http://cve.mitre.org/.

[Directive 1995] Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995, “*On the protection of individuals with regard to the processing of personal data*”, 1995.

[Demeanor *et al.* 2001] N. Demeanor, N. Delay, E. Lupus, M. Sloan. “The Ponder Policy Specification Language”, *International Workshop Policy*, Bristol, UK, IEE Computer Society Press, pp.18- 38, 2001.

[EC 1994] European Council, Bangemann report recommendations to the EC, 26 May 1994.

[Elkeelany 2002] M. Elkeelany, M.M. Matalgah, K.P. Sheikh, M. Thaker, G. Chaudhry, D. Medhi, J. Qaddour, "Performance analysis of IPSec protocol: encryption and authentication", *IEEE Communications Conference* (ICC 2002), pp. 1164–1168, ISBN: 0-7803-7400-2, 2002.

[Fitting 1993] M. Fitting, “*Basic Modal Logic*”, Handbook of Logic in Artificial Intelligence and Logic Programming Logic Foundations, (D.M. Gabbay, C.J. Hogger, J.A. Robinson, Eds.). Vol. 1/5, pp.365-448, ISBN 0-19-853745-X, Oxford Science Publications, 1993.

[Gad El Rab 2008] M. Gad El Rab, "*Évaluation des Systèmes de Détection d'Intrusions*",Thèse de doctorat, Université Paul Sabatier, 2008.

[Gad & Abou El Kalam 2006] M. Gad, A. Abou El Kalam, "Testing Intrusion Detection Systems: An Engineered Approach", 10th *Software Engineering Applications* (SEA), Dallas, États-Unis,13-15 novembre 2006.

[Gad *et al.* 2008a] M. Gad El Rab, A. Abou El Kalam, Y. Deswarte, "Execution Patterns in Automatic Malware and Human-Centric Attacks", IEEE *International Symposium on Network Computing and Applications* (IEEE NCA 2008)*,* 10 - 12 July 2008, Cambridge, MA USA, IEEE Computer Society.

[Abou El Kalam *et al.* 2008b] A. Abou El Kalam, M. Gad, Y. Deswarte, "Modélisation des processus d'attaques pour l'évaluation des IDS", Third Conference on Security in Network Architecture and Information Systems (SARSSI 2008), Loctudy, France | October 13 — 17, 2008.

[Gad *et al.* 2009] M. Gad El Rab, A. Abou El Kalam, Y. Deswarte, , "Manipulation of Network Traffic Traces for Security Evaluation", *IEEE International Workshop on Quantitative Evaluation of Large-Scale Systems and Technologies (IEEE QuEST 2009)*, Bradford, UK, 26-29 juin 2009, IEEE Computer Society.

[Hansmann 2003] S. Hansmann, "*A Taxonomy of Network and Computer Attacks*", Thèse de doctorat, University of Canterbury, New Zealand, 2003.

[Harkins] Harkins, D., Carrel, D., "The Internet Key Exchange (IKE)", RFC 2409, November1998.

[Harrison *et al.* 1976] M.A. Harrison, W.L. Ruzzo et J.D. Ullman, "Protection in Operating Systems", *Communication of the ACM,* 19(8), pp. 461-471, août 1976.

[Hentrayck 1989] P.V. Hentrayck, "Constraint Satisfaction in Logic Programming", MIT Press 1989.

[Hilty *et al.* 2007] M. Hilty, A. Pretschner, D. Basin, C. Schaefer and T. Walter, "A Policy Language for Distributed Usage Control", 12th *European Symposium On Research In Computer Security* (*ESORICS*), Dresden, Germany, September 24 26, 2007.

[Hou & Xia 2009] Delin Hou, Huosong Xia, "Design of Distributed Architecture Based on Java Remote Method Invocation Technology," *Environmental Science and Information Application Technology, International Conference on*, vol. 2, pp. 618-621, *International Conference on Environmental Science and Information Application Technology*, 2009.

[Howard 1998] J. D. Howard, "*An analysis of security incidents on the Internet 1989-1995*", Thèse de doctorat, Carnegie Mellon University, USA, 1998.

[IRGC 2007] International Risk Governance Council, "Critical infrastructures at risk: Securing the European electric power system", 2007.

[IANA 2009] Internet Assigned Numbers Authority (IANA), "port numbers", disponible à <http://www.iana.org/assignments/port-numbers>

[Iperf 2009] NLANR/DAST, Iperf tool, disponible à : <http://dast.nlanr.net/Projects/Iperf/>.

[Jaffar & Lassez 1987] J. Jaffar & J.L. Lassez, "Constraint logic programming", 14th *Annual ACM Symposium Principles of Programming Languages*, pp. 111-119, Munich, Allemagne, janvier 1987.

[Ken & Atkinson 98a] S. Kent et R. Atkinson. "Ip authentication header", RFC 2402, 1998.

[Ken & Atkinson 98b] S. Kent et R. Atkinson. "IP Encapsulating Security Payload (ESP)", RFC 2406, 1998.

[Ken & Atkinson 98c] S. Kent et R. Atkinson. "Security Architecture for the Internet Protocol", RFC 2401, 1998.

[Kendall 1999] K. Kendall, "A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems", *Master Thesis*, MIT Department of Electrical Engineering and Computer Science, USA, 1999.

[Keromytis 2003] A. D. Keromytis, J. L. Wright, T. Raadt, "The Design of the OpenBSD

Cryptographic Framework", *USENIX Annual Technical Conference*, General Track'2003. pp.181~196, 2003, disponible à : <http://www.openbsd.org/papers/ocf.pdf>.

[Kowalski 1974] R. Kowalski, "Predicate logic as programming language", *Information Processing*, 74, 569-574, J. L. Rosenfeld (ed.), North-Holland, 1974.

[Kripke 1963] S. Kripke, "Semantical Consideration in Modal Logic", *Acta Philosophical Logic*, vol. 16, 1963, pp. 83-94.

[Kumar 1995] S. Kumar, "Classification and Detection of Computer Intrusions", *PhD thesis*, Purdue University, USA, 1995.

[Lampson 1971] B. Lampson, "Protection", *5th Princeton Symposium on Information Sciences and Systems*, 1971.

[Lindqvist 1997] U. Lindqvist and E. Jonsson, "How to systematically classify computer security intrusions", *IEEE Symposium on Security and Privacy*, Oakland, CA, USA, pp. 154-163, 1997.

[Lippmann *et al.* 2000a] R. Lippmann, D. Fried, I. Graf, J. Haines, K. Kendall, D. McClung, D. Weber, S. Webster, D. Wyschogrod, R. Cunningham, M. Zissman, "Evaluating Intrusion Detection Systems: The 1998 DARPA Off-line Intrusion Detection Evaluation", *Proceedings of the DARPA Information Survivability Conference and Exposition (DISCEX 2000)*, Los Alamitos, CA, USA, pp. 12-26, 2000.

[Lippmann *et al.* 2000b] Richard Lippmann, Joshua W. Haines, David J. Fried, Jonathan Korba and Kumar Das, "Analysis and Results of the 1999 DARPA Off-Line Intrusion Detection Evaluation", *International Symposium on Recent Advances in Intrusion Detection (RAID* 2000*)*, Toulouse, France, Springer, LNCS 1907, pp. 162-182, 2000.

[Loi 2002] Loi 2002-303 sur les droits des patients et la qualité des soins, Article L. 1111-7, March 2002.

[Lupu &Sloman 1999] E. C. Lupu and M. Sloman, "*Conflicts in policy-based distributed systems management*" Soft. Eng., vol. 25, 1999.

[Moteff *et al.*, 2003] John Moteff, Claudia Copeland, and John Fischer, Critical Infrastructures: What Makes an Infrastructure Critical?, Rapport du "*Congressional Research Service*" No. RL31556, 2003, 20 pp. <http://www.fas.org/irp/crs/RL31556.pdf>

[Moteff & Parfomak 2006] John Moteff, Paul Parfomak, Critical Infrastructure and Key Assets: Definition and Identification, Rapport du "*Congressional Research Service*" No. RL32631, 2006, 19 pp. <http://fas.org/sgp/crs/RL32631.pdf>

[Moffett & Sloman 1993] J. D. Moffett and M. Sloman, "Policy Conflict Analysis in Distributed System Management," *Organizational Computing*, 1993.

[MGEN 2009] The Multi-Generator (Mgen), Naval Research Laboratory, disponible à: <http://cs.itd.nrl.navy.mil/work/mgen/>.

[Mostafa *et al.* 2008a] M. Mostafa, A. Abou El Kalam, C. Fraboul, "EESP: A Security Protocol that Supports QoS Management", *International Conference on Risks and Security of Internet and Systems*, Tozeur - Tunisia, 28-31 October 2008, IEEE.

[Mostafa *et al.* 2008b] M. Mostafa, A. Abou El Kalam, C. Fraboul, "A New QoS Controllable Security Protocol", *IEEE Annual Computer Security Application Conference*, (IEEE ACSAC'09),WiP,

Anaheim, USA, *08-12 décembre 2008*, IEEE Computer Society.

[Mostafa *et al.* 2009] M. Mostafa, A. Abou El Kalam, C. Fraboul, "Extending Firewall Session Table to Accelerate NAT, QoS Classification and Routing", 19th *IEEE International Conference on Computer Theory and Applications* (ICCTA'09), Alexandrie – Égypte, 17-19 octobre 2009.

[NERC 2003] North American Electric Reliability Council, "Urgent action standard 1200", 2003.

[NetBSD 2009] NetBSD 5.0 Release Candidate 2, février 2009., disponible à <ftp://ftp.netbsd.org/pub/NetBSDdaily/netbsd-5-0-RC2/ >.

[Ni *et al.* 2008] Q. Ni, E. Bertino, J. Lobo, "An Obligation model bridging access control policies and privacy policies", 13th *ACM symposium on access control models and technologies (SACMAT*), Estes Park, CO, USA, June 11-13, 2008.

[Nichols *et al.* 1998] K. Nichols, S. Blake, F. Baker, D. Black, "Definition of the Differentiated Services Field in the IPv4 and IPv6 Headers". RFC 2474, December 1998.

[OASIS, 2003] OASIS, XACML Specification V1.1, 24 juillet 2003. <www.oasis-open.org/committees/xacml/repository/cs-xacml-specification-1.1.pdf>

[OASIS, 2005] OASIS, "UDDI Specifications TC, Universal Description, Discovery and Integration", v3.0.2, Février 2005. <http://www.uddi.org/pubs/uddi_v3.htm>.

[OSVDB 2008] Open Source Vulnerability Databse-OSVDB (2008): http://osvdb.org/.

[Prior 1954] A. Prior, "The paradoxes of derived obligation". Mind vol. **63** (1954) 64–65.

[Resolution 1990] Resolution A/RES/45/ de l'Assemblée générale des Nations Unies, Guidelines for the regulation of computerized personal data files, Déecembre 1990.

[Recommandation 1992] Recommandation du conseil de l'Europe, "Communication of Health Information in Hospitals", European Health Committee CDSP (92)8, Strasbourg, juin 1992.

[Recommandation 1997] Recommandations du conseil de l'Europe, R(97)5, "On The Protection of Medical Data Banks", Council of Europe, Strasbourg, 13 février 1997.

[Recommandation 1995] Recommandation no R (95) 4 du conseil de l'Europe sur la "protection des données à caractère personnel", Strasbourg, 7 février 1995.

[RFC 2119] S. Bradner. "RFC2119: Key words for use in RFCs to Indicate Requirement Levels", IETF, Mars 1997.

[RFC3198] RFC 3198, *Terminology for Policy-Based Management*, Internet Society, November 2001, disponible à : <http://www.faqs.org/rfcs/rfc3198.html>

[Samuel 2003] Samuel J. Leffler, Errno Consulting. Fast IPSec: A High-Performance IPsec Implementation. *USENIX Association, Proceedings of BSDCon*'03. San Mateo, États-Unis. 8–12 septembre 2003, pp. 133-140.

[Sandhu *et al*. 2000] R.S. Sandhu, D. Ferraiolo et R. Kuhn, "The NIST Model for Role-Based Access Control: Towards a Unified Standard", in *5th ACM Workshop on Role-Based Access Control,* Berlin (Allemagne), pp. 47-63, 2000.

[Schneider 1994] B. Schneider, "Description of a new Variable-Length Key, 64-Bit Block Cipher", *Software Encryption, Cambridge Security Workdhop*, Springer-Verlag, 1994, pp. 191-204.

[Solms & Merwe 1994] S.H. von Solms et I. Van der Merwe, “The Management of Computer Security Profiles Using a Role-Oriented Approach”, *Computer & Security*, vol.13, n° 8, pp. 673-680,1994.

[Szigeti & Hattingh 2004] T. Szigeti, C. Hattingh, “End-to-End QoS Network Design: Quality of Service in LANs, WANs, and VPNs”. *Cisco Press*, 1st edition, 2004, ISBN 1-58705-176-1.

[Strembeck 2004] M. Strembeck, “Conflict Checking of Separation of Duty Constraints in RBAC” *Conference on Software Engineering*, 17-19 février 2004, Innsbruck, Austria.

[Suominen 2004] K. Suominen, NetBSD 5.0 Manual Pages, Mars 2004, disponible à <http://netbsd.gw.com/cgibin/man-cgi?setkey++NetBSD-current>.

[Sun 2006] Sun microsystems, “*Java Remote Method Invocation*”, documentation officielle disponible à <http://java.sun.com/javase/6/docs/technotes/guides/rmi/index.html>

[Toman 1997] David Toman, "Memoing Evaluation for Constraint Extensions of Datalog", *Constraints*, 3(3/4) ; 337-359, 1997.

[Verìssimo *et al.* 2008] P. Verìssimo, N.F. Neves, M. Correia, Y. Deswarte, A. Abou El Kalam, A. Bondavalli, A. Daidone, “The CRUTIAL Architecture for Critical Information Infrastructures”, “5*th Architecting Dependable Systems (ADS) Book*”, ISBN: 978-3-540-74033-9, 2008, Springer, http://www.springerlink.com/content/978-3-540-85570-5/.

[W3C, 2003] W3C, “SOAP, Version 1.2” W3C Recommendation, juin 2003. http://www.w3.org/TR/soap12-part0/>

[W3C, 2004] W3C, “Extensible Markup Language (XML)”, W3C Recommendation, février 2004. <http://www.w3.org/XML/>.

[W3C, 2006] W3C, “Web Services Description Language (WSDL), Version 2.0”, W3C Candidate Recommendation, mars 2006. <http://www.w3.org/TR/wsdl>

[Weber 1998] D. J. Weber "*A taxonomy of computer intrusions*", Thèse de Master, Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science, USA, 1998.